



MP MASTER



MA32764



Summit Series
Probe Station Communications
Programming Guide V2.0



Cascade Microtech, Inc., 14255 SW Brigadoon Ct., Beaverton, Oregon 97005
Tel: (503) 626-8245 Fax: (503) 626-6023
Japan: (03) 3320-6410; Europe: +44 (0) 295-721216
Copyright © 1995 Cascade Microtech, Inc.

Summit Series
Probe Station Communications
Programming Guide

106-910-B

List of Trademarks and Registered Trademarks

Cascade Microtech is a registered trademark of Cascade Microtech, Inc.

Hewlett-Packard is a registered trademark of Hewlett-Packard Company.

Windows is a trademark of Microsoft Corporation.

Copyright Notice

© Copyright 1995 Cascade Microtech, Inc. All rights reserved. No part of this manual may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopy, recording, or any information storage and retrieval system, without permission in writing from Cascade Microtech, Inc.

Requests for permission to make copies of any part of this manual should be mailed or faxed to:

Cascade Microtech, Inc.
14255 SW Brigadoon Ct.
Beaverton, Oregon 97005 USA
Tel: (503) 626-8245
Fax: (503) 626-6023

Warranty

Cascade Microtech, Inc. (Cascade) warrants each product it sells to be free from defects in material and workmanship under normal use and service. Software and firmware is warranted to operate in accordance with its programmed instructions on appropriate Cascade products; it is not warranted to be error-free.

During the warranty period, Cascade will, at its option, either repair or replace the product.

For warranty service or repair, this product must be returned to a service facility designated by Cascade. Cascade assumes no risk for damage in transit. Buyer shall prepay shipping charges to Cascade and Cascade shall pay shipping charges to return the product to buyer. However, buyer shall pay all shipping charges, duties, and taxes for products returned to Cascade from another country.

Refer to the product specifications information in the data sheet, product warranty list, or product manual for the warranty period for the product. The warranty begins on the date of shipment.

Limitation of Warranty

The foregoing warranty shall not apply to effects resulting from improper or inadequate maintenance by buyer, buyer-supplied software or interfacing, unauthorized modification or misuse, operation outside the environmental specification for the product, or improper site preparation and maintenance.

Exclusive Remedies

The foregoing warranty is in lieu of all other warranties, express or implied, including, but not limited to the implied warranties of merchantability and fitness for a particular purpose, which are hereby disclaimed. In no event will Cascade Microtech, Inc. be liable for consequential damages, even if Cascade Microtech, Inc. has been advised of the possibility of such damages.

A B O U T T H I S G U I D E

Introduction

Welcome to the *Probe Station Communications Guide*. This communications guide explains how to control your probe station from a remote host, or from another Windows application running on the probe-station computer. With the information in this guide, you can write automatic test programs that control the probe station. All Cascade Microtech Summit-series semi-automatic probe stations use the same software interface.

Organization

You do not have to read this guide from front to back. Each chapter in this guide documents a particular type of communication:

For	Refer To Chapter
An overview of probe station communications	1: <i>Starting Out</i>
The Dynamic Data Exchange Program Interface	2: <i>DDE Server Communications</i> 4: <i>Command Dictionary</i> <i>see also</i> DDE online help
Remote Communication Over GPIB	3: <i>GPIB Communications</i> 4: <i>Command Dictionary</i> <i>see also</i> GPIB online help
Command Server programmatic interface	4: <i>Command Dictionary</i> 5: <i>Command Server Communications</i>

Additional Information

Other sources of information that you might find useful include:

- *Prober Control Software User Guide* and online help
- DDE server online help
- GPIB server online help
- User guide for your probe station
- *WinCal User Guide* and online help
- Visual Basic Examples. Along with your PCS software, you should have received a disk of programming examples written in Visual Basic. These examples were designed to address some of the more common testing requirements. Copy and

modify them, as needed, to meet your testing needs. These programs are documented in the *VBREADME.TXT* file.

Update and Revision Information

Cascade tries to provide you with documentation that is complete and up-to-date. When features are added or revisions made after the manual is printed, the information is provided along with the software in a *readme.txt* file.

CONTENTS

ABOUT THIS GUIDE	i
Introduction.....	i
Organization	ii
Additional Information	ii
Update and Revision Information.....	iii
 STARTING OUT	 1-1
Introduction.....	1-1
Before You Begin.....	1-2
System Requirements	1-2
Technical Support	1-3
Using Online Help.....	1-4
The Help Menu.....	1-4
Context-Sensitive Help.....	1-5
The Help Window	1-5
How Probe Station Communication Works	1-6
 DDE SERVER COMMUNICATIONS	 2-1
Introduction.....	2-1
About DDE.....	2-1
Basic Concepts and Terminology	2-2
Client/Server Conversation.....	2-2
Application, Topic, and Item.....	2-3
Type of Conversation.....	2-3

Starting DDE	2-5
Main Steps for Using DDE.....	2-6
Displaying Data	2-7
Selecting and Running Your Application.....	2-8
Controlling the Probe Station Directly.....	2-10
Operating in Virtual Mode.....	2-11
Exiting DDE	2-11
DDE Server Commands	2-11
DDE Programming Examples.....	2-12
C Example.....	2-12
Visual Basic Example.....	2-15
GPIB COMMUNICATIONS.....	3-1
Introduction.....	3-1
About the GPIB Server.....	3-1
GPIB Concepts and Terminology.....	3-2
GPIB and SCPI Standards	3-2
Instrument vs. Controller.....	3-3
Command and Query Responses	3-4
Setting Up for Remote Communication	3-5
Setting GPIB Communication	
Parameters.....	3-5
Verifying GPIB Communication.....	3-8
Identifying GPIB Communication	
Problems	3-11
Handshaking, Status Reporting, and Time-outs	3-12
Status-Reporting Structure.....	3-13
GPIB Timeout Values.....	3-17
Using Serial Polls	3-20
Using Service Requests (SRQs)	3-23
Using String-Return Handshaking	3-26
Understanding Command Execution	3-27
Types of Commands.....	3-29
Probe Station Commands	3-29
Meta Commands.....	3-30
GPIB Protocol Commands	3-33
Comments and Messages in Programs	3-35
Advanced Troubleshooting	3-35

Introduction.....	4-1
Command Syntax	4-2
Summary of Commands	4-4
:move:absolute	4-20
:move:absolute?.....	4-23
:move:absolute:chuck	4-25
:move:absolute:nchuck.....	4-28
:move:center.....	4-31
:move:contact	4-33
:move:contact?	4-35
:move:down.....	4-37
:move:down?	4-39
:move:home	4-41
:move:load.....	4-43
:move:probeplan:absolute:die	4-45
:move:probeplan:absolute:die?	4-47
:move:probeplan:absolute:index	4-49
:move:probeplan:absolute:index?	4-51
:move:probeplan:absolute:subsite.....	4-53
:move:probeplan:absolute:subsite?	4-56
:move:probeplan:first:die	4-58
:move:probeplan:first:subsite	4-61
:move:probeplan:last:die	4-64
:move:probeplan:last:subsite	4-66
:move:probeplan:next:die	4-68
:move:probeplan:next:site	4-70
:move:probeplan:next:subsite	4-72
:move:probeplan:prior:die	4-74
:move:probeplan:prior:site	4-76
:move:probeplan:prior:subsite	4-78
:move:relative.....	4-80
:move:relative:chuck	4-83
:move:relative:nchuck	4-86
:move:scan:aaxis	4-89
:move:scan:xaxis	4-92
:move:scan:yaxis	4-94
:move:scan:zaxis	4-97
:move:separate	4-99

:move:separate?	4-101
:move:stop:aaxis	4-103
:move:stop:xaxis	4-105
:move:stop:yaxis	4-107
:move:stop:zaxis	4-109
:move:up	4-111
:move:up?	4-113
:move:wait	4-114
:move:wait?	4-116
:probeplan:align	4-118
:probeplan:delete	4-120
:probeplan:download	4-122
:probeplan:load	4-125
:probeplan:load?	4-128
:probeplan:mark:die	4-130
:probeplan:nsubsites?	4-133
:probeplan:ntested?	4-135
:probeplan:set:palette	4-137
:probeplan:show	4-140
:probeplan:set:subsite	4-142
:probeplan:unload	4-145
:probeplan:upload	4-147
:set:acceleration	4-149
:set:auxiliary	4-151
:set:busy?	4-152
:set:compensation	4-153
:set:compensation:factor	4-155
:set:compensation:mode	4-157
:set:delay	4-159
:set:delay?	4-161
:set:display:errors	4-163
:set:display:messages	4-165
:set:edgesense	4-167
:set:edgesense:overdrive	4-169
:set:edgesense:switch	4-171
:set:host	4-173
:set:host?	4-175
:set:light	4-177

:set:manual	4-178
:set:manual?	4-180
:set:msafety	4-182
:set:msafety?	4-184
:set:precise	4-185
:set:preset	4-186
:set:priority:accuracy:module	4-188
:set:priority:accuracy:wafer	4-190
:set:priority:standard:module	4-192
:set:priority:standard:wafer	4-194
:set:quiet	4-196
:set:salign	4-197
:set:salign?	4-199
:set:show:command	4-200
:set:show:command?	4-202
:set:show:error	4-204
:set:show:error?	4-206
:set:show:response	4-208
:set:show:response?	4-210
:set:station?	4-212
:set:time:command	4-213
:set:time:command?	4-215
:set:time:error	4-217
:set:time:error?	4-219
:set:travel	4-221
:set:travel?	4-224
:set:unit	4-226
:set:vacuum	4-228
:set:velocity	4-230
:set:virtual	4-232
:set:virtual?	4-234
:set:wafer	4-236
:system:beep	4-238
:system:delay	4-239
:system:display	4-241
:system:error?	4-243
:system:identification?	4-245
:system:lock	4-246

:system:lock?.....	4-247
:system:path?	4-249
:system:unlock.....	4-250
:system:version?	4-252

COMMAND SERVER COMMUNICATIONS 5-1

Introduction.....	5-1
About the Command Server.....	5-2
Command Server Program Example	5-2
CMDemo.C Listing	5-3
CMDemo.H Listing	5-9
CMDemo.RC Listing.....	5-10

Index I-1

Figures

Figure 1-1. Online Help screen.....	1-6
Figure 1-2. Probe-station communications model.....	1-8
Figure 2-1. Client initiates conversations	2-4
Figure 2-2. DDE Server always returns responses	2-4
Figure 2-3. Client terminates DDE Server conversations.....	2-5
Figure 2-4. Dynamic Data Exchange server window	2-6
Figure 2-5. Setup Options dialog box	2-7
Figure 2-6. Execute Commands dialog box.....	2-8
Figure 2-7. Open Command Files dialog box.....	2-9
Figure 2-8. Enter Command dialog box	2-10
Figure 3-1. Test environment	3-2
Figure 3-2. GPIB Server window	3-6
Figure 3-3. GPIB Setup dialog box	3-7
Figure 3-4. GPIB Server window when server is online.....	3-9
Figure 3-5. Status-reporting structure.....	3-14
Figure 3-6. Sample Poll_Until_Done routine	3-22
Figure 3-7. Turning on, waiting for, and turning off SRQs.....	3-24
Figure 3-8. CSGPIB.INI file	3-36
Figure 3-9. GPIBMOVE.GPB file.....	3-38

Tables

Table 3-1. Recommended Meta-Command Settings ^a	3-27
Table 3-2. Remote-Mode Meta Commands ^a	3-31
Table 3-3. GPIB Protocol Commands	3-33
Table 4-1. Abbreviated Commands	4-3
Table 4-2. Prober Control Commands ^a	4-4

1

STARTING OUT

Introduction

Cascade Microtech provides five software interfaces for controlling Summit-series semi-automatic probe stations:

Refer to the
*PCS User's
Guide* for
PCS
information.

- Prober Control Software (PCS), which is menu-based, interactive software that runs under Microsoft Windows.
- DDE Server to enable you to use Windows application programs running on the probe-station computer to control the hardware.
- GPIB Server to give you remote control over the probe station.
- Command Server to allow your application programs to bypass the DDE Server and interact directly with the probe station.

- Summit Software, which is menu-based, interactive software that runs under MS DOS.

This guide discusses the DDE, GPIB, and Command Servers.

This section contains the following information:

- system requirements
- technical support
- using online help
- an overview of probe-station communications

Before You Begin

If you are familiar with navigating and using the mouse in Microsoft Windows, you will be able to use the DDE and GPIB Server menus. If you need to know more about these topics, please refer to your Microsoft Windows documentation.

System Requirements

The DDE, GPIB, and Command Servers work with the following *minimum* system requirements:

- A 386-class IBM-compatible computer with a standard AT-bus
- DOS Version 6.0
- Windows Version 3.1
- Mouse
- 4 MB RAM
- Color SVGA monitor
- One high-density, 3.5-in. floppy disk drive
- One hard disk drive with approximately 5 MB available disk space

Technical Support

Call 503/626-8245 and ask for customer service.

If you need technical support, please read everything relevant to the problem in this guide and in the online help before calling customer service.

To receive a faster solution, try to re-create the problem to provide us with an exact sequence of events. Please have the following information available, if possible:

- ✓ Name, version number, and file date of the application in use.
- ✓ Disk drive information, including sizes, hard drive controller card brand, partition sizes, and partitioning software.
- ✓ Additional hardware such as specialty video cards, EMS boards, or turbo cards.
- ✓ Memory-resident programs in use when the problem occurred. Problems can occur when memory-resident programs are not loaded in the correct order.
- ✓ Contents of your AUTOEXEC.BAT and CONFIG.SYS files.
- ✓ Exact wording of any error messages.
- ✓ Notes about any steps you took in trying to solve the problem, and their outcome.
- ✓ MS-DOS version number and manufacturer.
- ✓ MS-Windows version number and manufacturer.

Please review your AUTOEXEC.BAT file.

Using Online Help

Click Help in the menu bar.

Both the DDE and GPIB Server applications provide online Help that describes the menu interface, including command descriptions similar to what is available in the *Command Dictionary* chapter of this manual.

The Help Menu

To access online help, click Help in the DDE Server or GPIB Server window. The Help menus contain these commands:

Select	To Do This
Contents	View the table of contents for the server. To view help for a particular topic, click the underlined word or phrase.
Command List	View a list of commands. To view help for a particular command, click the underlined word or phrase.
Search for Help on	Search for help on a specific topic. Type a topic in the text box and press Enter or choose Show Topics . A list of possible help topics appears. To view help for a particular topic, select the topic you want and choose Go To .
How to Use Help	Learn more about using the Microsoft Windows help system. To view help for a particular topic, click the word or phrase.

About ...

View information about the server.
The window that appears displays the application name, version number, and copyright date.

Context-Sensitive Help

In dialog boxes, you can click the **Help** button to access specific information about the dialog-box options.

The Help Window

Throughout the help system, you can click any word with a solid underline to go directly to a help screen for that topic. Or, you can click a word or phrase with a dotted underline to see pop-up help for that topic or term.

The following sample help window illustrates the features of the online help system:

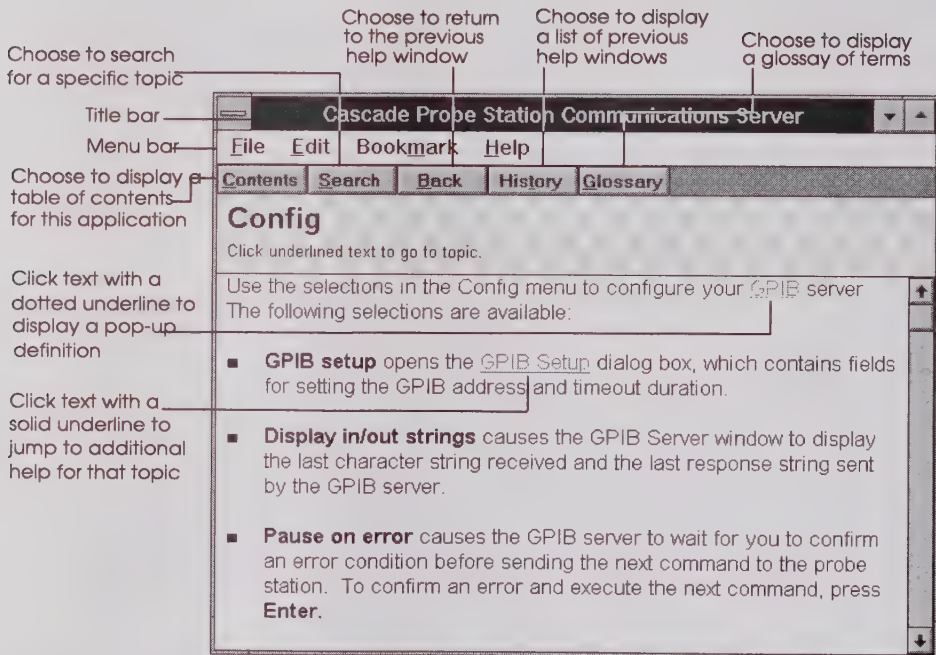


Figure 1-1. Online Help screen

How Probe Station Communication Works

The Command Server software module accepts commands from other applications, sends them to the probe-station drivers, and returns responses.

Chapter 4 of this guide contains command descriptions.

To the Command Server (CS), the DDE Server, the GPIB Server, and test programs are equivalent client applications. It acts as the intermediary between its clients and the probe-station drivers to translate the commands.

A DDE application program needs to use DDE-specific commands to open communication with the DDE Server. Once that is accomplished, the program sends probe-station control

commands to the DDE Server, which passes them to the CS. The CS sends the commands to the probe-station's hardware driver software, which executes the commands on the probe station. Responses from the hardware are sent back through the CS and the DDE Server to the DDE application program.

If desired, you can write an application program that bypasses the DDE Server and communicates directly with the CS.

The GPIB Server receives commands from a remote host over the GPIB and passes them to the CS, which in turn passes them on to the probe-station hardware driver. The probe station returns responses to the CS, which passes them to the GPIB Server for sending out over the GPIB to the remote host.

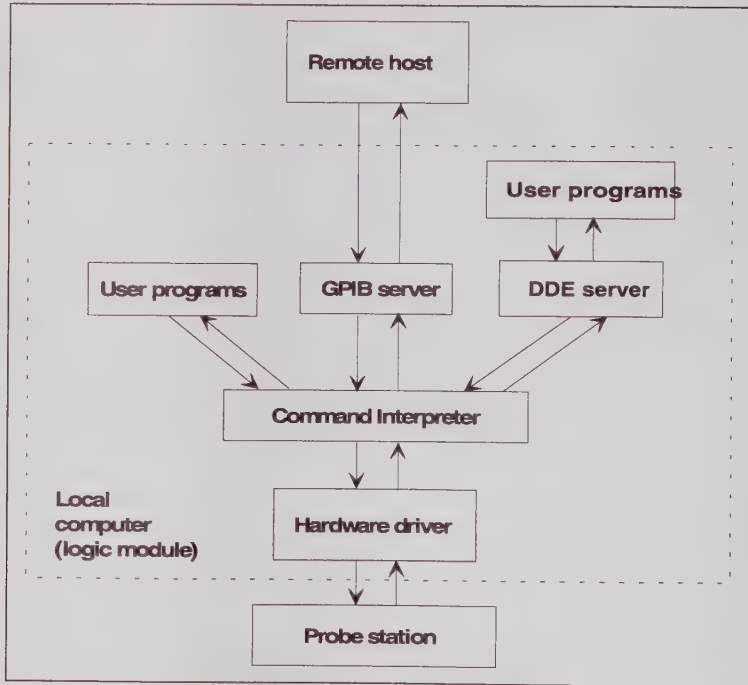


Figure 1-2. Probe-station communications model

From an application programmer's point of view, the DDE commands, the GPIB commands, and the CS that control the probe station are virtually identical. Both command sets use the same syntax. The only difference is that there are some commands that are useful only in a GPIB session.

In addition to Cascade's library of prober control commands, the GPIB Server recognizes commands that control the bus (see Chapter 3). And, the DDE Server recognizes the commands in the Windows DDE Management Library.

2

D D E S E R V E R C O M M U N I C A T I O N S

Introduction

The Dynamic Data Exchange (DDE) Server application allows you to write automated test programs to control your probe station. The DDE Server accepts commands from other Windows applications, sends them to the probe station by way of the Command Server, and returns responses.

This section contains the following information:

- overview of DDE
- concepts and terminology used with MS-Windows DDE applications

About DDE

The Cascade Dynamic Data Exchange (DDE) Server provides you with interprocess communication. It gives you the ability to control the probe-station hardware from other Windows applications. When you activate the DDE Server, it opens communication with the Command Server, initializes any Cascade hardware attached to the probe station, displays a menu for local-control options, and waits to receive a DDE command from another application.

The end of this section lists DDE program examples.

When writing programs to communicate with the DDE Server, you can use the Windows DDE Management Library (DDEML). Using the library simplifies DDE programming by providing high-level function calls and by letting programs respond to DDE messages with a call-back function.

Basic Concepts and Terminology

Refer to your Windows programming documentation for DDE Management Library (DDEML) information.

The DDE application is based on the DDE messaging system built into Windows. The concepts and terms presented here are common to all DDE applications.

Client/Server Conversation

Communication with the DDE Server follows the client/server model. Essentially, two Windows programs carry on a "conversation" by posting messages to each other. These two programs are known as the "server" and the "client". A server, such as DDE, is the program that has access to data that the other program needs. A client is the program that requests and receives data from the server.

The client program initiates DDE conversations. The client broadcasts a message to all running Windows programs that

indicates the type of information it needs. The server that has the information, in this case DDE, responds and the conversation begins.

Application, Topic, and Item

When a client asks a server for data, it uses three character strings to identify the type of data that it wants:

To see these strings in use, refer to the sample program later in this section.

1. The server application name, which is the name of the program. For DDE the application string is

"EDMAIN"

2. The topic name, which identifies the type of data. DDE has only one topic:

"CASCADE Commands"

3. The item name. The items in DDE are the actual commands. For example, **:move:load 2**.

Type of Conversation

Refer to your Windows programming documentation for more information.

Windows supports three basic types of interprocess conversations: cold link, hot link, and warm link. DDE uses the cold link (or execution).

A DDE conversation begins with the client broadcasting an initiate message. The server responds to the client with an acknowledge message, as shown in Figure 2-1.



Figure 2-1. Client initiates conversations

The client then makes function calls to the DDE Server using the Windows DDEML's `DdeClientTransaction()` Application Program Instruction (API). The transaction type is set to `XTYP_EXECUTE` and the item parameter is set to point to a null-terminated string containing the command to be executed (See the programming example listed near the end of this section).

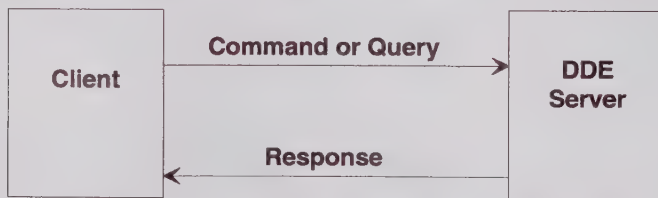


Figure 2-2. DDE Server always returns responses

For every client call, the DDE Server uses the `DdeClientTransaction()` return parameter to return a pointer to a status string containing one of the following values:

- A pointer to a string containing "COMPLETE", indicating that the transaction executed without error.
- A pointer to a string containing the answer to a query made by the client. The format of the string depends on the nature of the requested data.

- A pointer to a string containing an error message. The first character of any error-message string is the symbol @. This is followed by the error message, a new line character combination, and the command string that generated the error condition.

The conversation ends when the client posts a disconnect messages to the server, as shown in Figure 2-3. The DDE Server disconnects without returning an acknowledge message:

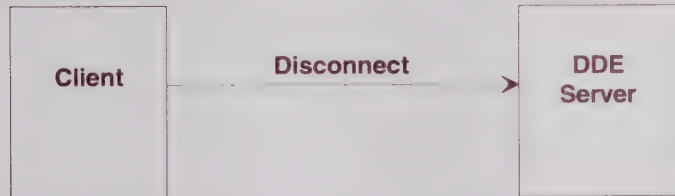


Figure 2-3. Client terminates DDE Server conversations

Starting DDE



When you installed the Cascade Prober Control Software (PCS), the Install program created a Cascade program group on the Program Manager desktop and placed DDE in that group.

From the desktop, double-click the DDE icon in the Cascade window. This initializes the probe-station hardware and displays the DDE window, shown in Figure 2-4.

You can also launch the DDE Server from the PCS window.



Figure 2-4. Dynamic Data Exchange server window

To start the DDE Server, click on Activate DDE in the DDE window's menu bar. This enables the DDE Server to accept DDE commands from other applications.

Main Steps for Using DDE

Most of the effort in using DDE is in developing your application program. The main steps for using DDE once you have written your application program are as follows:

Step	What You Do
1	In the Options dialog box, select the type of information you want to see in the DDE window during your program execution.
2	Select and run your DDE client application program.

Displaying Data

Select Setup to display the Options dialog box.

The lower portion of the DDE window is blank unless you select a type of data to display during the execution of your application program. You can choose the type of data by checking the boxes in the Setup Options dialog box, shown in Figure 2-5. The following choices are available:

Choose	To Do This
Display commands	View the probe-station commands as they are received by the DDE Server.
Display errors	View errors as the DDE Server generates them.
Display responses	View the messages the DDE Server is returning to the application program.
Cmd/Response	Set the display duration for commands and responses.
Errors	Set display duration for error messages.

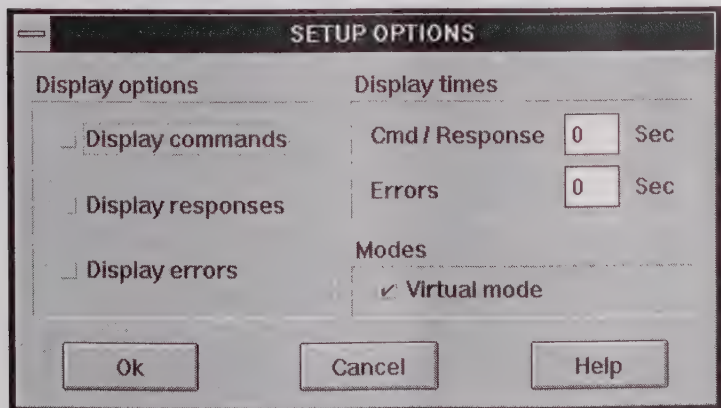


Figure 2-5. Setup Options dialog box

Selecting and Running Your Application

Once you've developed an application program to control your probe station, you can run it from the DDE Server window:

1. Click on **Stored** in the DDE Server window's menu bar. The screen displays the Execute Commands dialog box, shown in Figure 2-6.

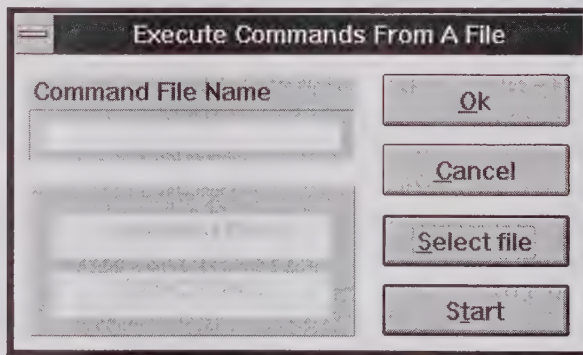


Figure 2-6. Execute Commands dialog box

2. Click on the **Select file** button in the Execute Commands dialog box. The screen displays the Open Command Files dialog box, shown in Figure 2-7.
3. Select the drive, directory, and file name of your application program file, then click on **OK**.

DDE Server command files should have a .DDE file extension.

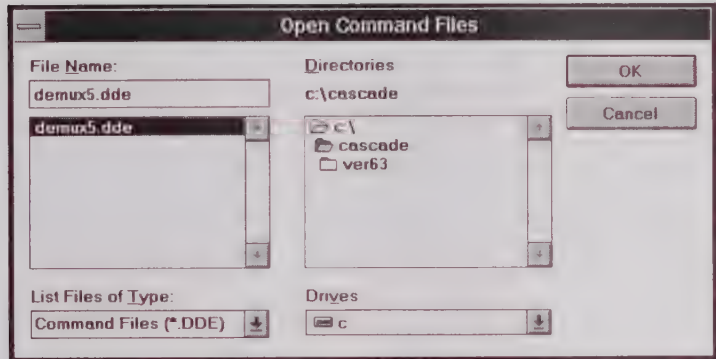


Figure 2-7. Open Command Files dialog box

4. Click on **Start** in the Execute Commands dialog box. The words Command file is EXECUTING are black while the application program is running and gray after the execution is complete.
5. Click on **OK** to close the Execute Commands dialog box while saving the command file selection. If you click on **Cancel**, the dialog box closes without saving your latest command-file selection.

If you do not want to run an application program, you can communicate directly with the DDE Server by sending commands individually. Refer to *Controlling the Probe Station Directly* for more information.

Controlling the Probe Station Directly

If you want to send an individual command to the probe station, you can do so by setting the probe station to operate in local mode. To do this, follow these steps:

1. Select **Direct** in the menu bar. The system displays the Enter Command dialog box, shown in Figure 2-8.
2. Type in the command or select it from the drop-down list box.
3. Click **OK**.

The DDE Server sends the command to the probe station.

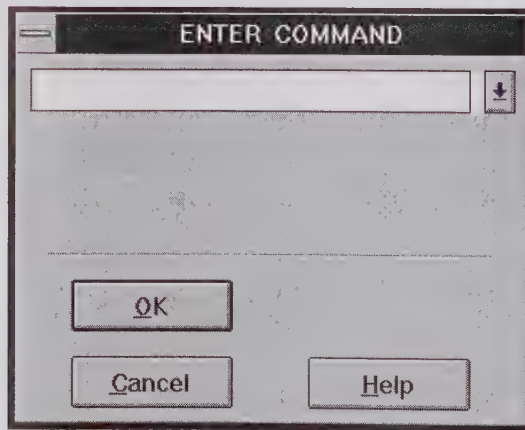


Figure 2-8. Enter Command dialog box

Operating in Virtual Mode

DDE gives you the ability to run programs without any probe-station hardware. This is useful for setting up demonstrations or for developing your application programs.

To operate in virtual mode, click on Setup and check **Virtual mode** in the Setup Options dialog box.

Exiting DDE

To end your DDE session:

1. Choose Exit from the File menu. The screen displays a dialog box asking you to confirm that you want to halt DDE communication.
2. Click on **Yes** to stop the execution of the DDE Server and to close the DDE window.

DDE Server Commands

Chapter 4 contains the command dictionary.

DDE Server commands enable your test programs to communicate with the probe station, providing you with an alternative to using the interactive menu system. Use Cascade's library of DDE commands in conjunction with the Windows DDE Management Library (DDEML) to automate your test and measurement tasks.

Refer to Chapter 5 for Command Server information. The DDE Server command syntax matches the command syntax for Cascade's Command Server commands. Chapter 4 of this manual outlines this command syntax and describes the commands.

Refer to
Chapter 3 for
GPIB Server
information.

The Standard Commands for Programmable Instruments (SCPI) standard describes a set of commands and a syntax for instrument communication. The SCPI command syntax is built upon the GPIB communication protocol. Cascade's DDE Server and Command Server commands comply with the SCPI command syntax.

DDE Programming Examples

This section contains two code fragments to help familiarize you with using the DDE Server. Your PCS installation disk contains additional program examples.

C Example

The following code fragment shows how to make a function call to the DDE Server. The application program is written in the C programming language, using Borland International's C compiler and ObjectWindows.

The call to the server uses the Windows application program instruction (API) `DdeClientTransaction()` to send a **:move:absolute** command. The transaction type is set to `XTYP_EXECUTE` and the DDE `hpszItem` parameter is set to point to a null-terminated string containing the command.


```
/* The following code fragment uses Windows DDEML
APIs to

send a move command to the DDE Server and accepts
back

a status string */
.
.
.

/* Set up the DDEML call back function that is used
by the

DDEML to carry out data transfers between
applications */

lpCallBack =
MakeProcInstance((FARPROC)TDdeClientWin::CallBack,

GetApplication()->hInstance);
BOOL err = (lpCallBack == NULL);

if(!err) err = DMLERR_NO_ERROR !=
DdeInitialize(&idInst,

(PFNCALLBACK)lpCallBack, APPCMD_CLIENTONLY, 0);

/* Set up the client server connection */

if(!err) {

hszService = DdeCreateStringHandle(idInst, "CMI PCS
Server",

CP_WINANSI);

hszTopic = DdeCreateStringHandle(idInst, "CMI
Commands",

CP_WINANSI);

hszItem = DdeCreateStringHandle(idInst,

":!::MOVE:ABSOLUTE 12345 54321 1",CP_WINANSI);

err = (hszService == NULL || hszTopic == NULL ||
```

```
hszItem == NULL):  
}  
if(!err)err = !(hConv = DdeConnect(idInst,  
hszService,  
  
hszTopic, NULL));  
if(err) {  
MessageBox(HWINDWO, "Unable to make DDE  
connection.\nMake  
sure WinPCS SERVER is running.", "DDE CLIENT  
ERROR",MB_ICONSTOP);  
PostQuitMessage(0);  
}  
  
/* Set up variables */  
HDEDATA transResult = NULL;  
DWORD dwResult = 0;  
char result[LONG_STR_LEN] = "Couldn't communicate  
with server";  
  
/* Need long timeout value to allow the station to  
respond */  
const long TIMEOUT = 60000;  
  
/* Send the command string to the server */  
transResult = DdeClientTransaction(NULL, 0, hConv,  
hszItem,  
CF_TEXT, XTYP_REQUEST, TIMEOUT_TIME, &dwResult);  
  
/* Read the result string. transResult will be a  
valid data  
handle if the client transaction above was  
successful, and  
NULL if it was not. This must be checked since  
calls to  
DdeGetData with a NULL handle cause GPF's. */
```

```
if(transResult) DdeGetData(transResult,
(LPBYTE)result, sizeof(result), 0);

/* Display the result string */
MessageBox(GetActiveWindow(), result, "RESULT",
MB_ICONINFORMATION);
.
.
.
```

Visual Basic Example

The following subroutine uses Visual Basic 3.0 to create a message box that you can use to send commands directly to the DDE Server.

```
-----  
Sub Form_DblClick ()  
    ' Controlling Summit Station using DDE  
    ' Visual Basic 3.0 Example  
    '-----  
    '  
    ' Steps to duplicate:  
    ' 1. Create a form with a single text box labeled  
text 1  
    ' 2. Create this routine  
    '  
    ' Run the program. Type a command into the text box.  
    ' Double-click on the form (outside the text box) to  
    ' send the command to the Probe Station Software.  
    ' The text box contents are replaced by the response  
    ' from the station.  
    '  
    '  
    If text1.LinkMode = 0 Then  
        text1.LinkTopic = "EDMAIN|COMMANDS"  
        text1.LinkItem = text1.Text  
        text1.LinkMode = 2  
    End If  
    text1.LinkItem = text1.Text  
    text1.LinkRequest  
End Sub
```

GPIB COMMUNICATIONS

Introduction

Cascade semi-automatic probe stations support remote communication across the General Purpose Interface Bus (GPIB). This feature allows you to send commands and information between the probe station and a remote host. The probe station remote-communication features help you integrate the probe station into an automated test environment.

About the GPIB Server

Requires probe station with GPIB board installed.

For communication across the GPIB, Cascade semi-automatic probe stations have both talker and listener GPIB capabilities. If you have a remote host acting as a GPIB controller, you can send and receive instrument settings across the GPIB. Figure 3-1 shows an example of several instruments which might connect to a host through a GPIB.

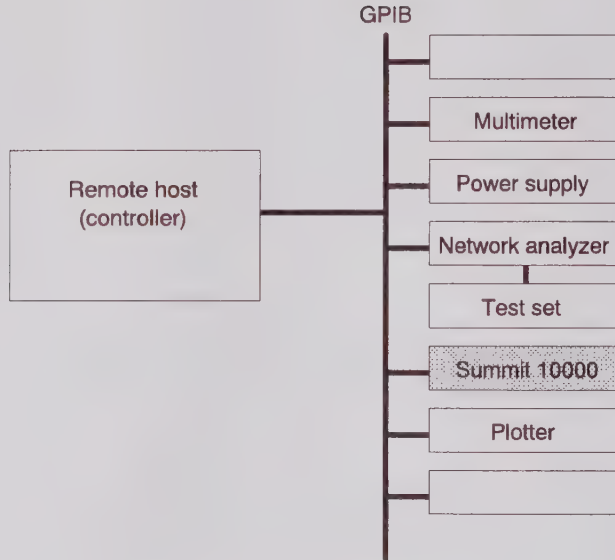


Figure 3-1. Test environment

GPIB Concepts and Terminology

GPIB and SCPI Standards

The GPIB is described in IEEE 488.1-1987 and 488.2-1987

The GPIB standard describes a method for allowing instruments to communicate across a bus. This includes a standard method for specifying which instrument is the talker, listener, and/or controller of the communication. This also includes a standard method for defining and describing status-reporting structures. The GPIB standard is fully described in the IEEE 488.1-1987 and 488.2-1987 documentation, published by IEEE.

Cascade probe stations support commonly used IEEE 488.1-1987 and 488.2-1987 commands such as status-reporting commands.

The probe station can communicate with other instruments both as a GPIB talker and listener.

The Standard Commands for Programmable Instruments (SCPI) standard describes a set of commands and a syntax for instrument communication. The SCPI commands and syntax are built upon the GPIB communication protocol. The SCPI standard is issued and controlled by the SCPI consortium.

Cascade probe-station commands comply with SCPI syntax and protocols but are not defined in the SCPI standard, because the standard does not include commands for probing equipment.

There are three Cascade command sets: GPIB commands, meta commands and the probe-station control commands. The GPIB command set reads and sets registers in the GPIB status reporting structure. Meta commands specify system-level functions such as error reporting. And, the probe-station control command set includes commands for translating movement along the x-, y-, and z-axes, for managing files, and for setting up the probe station. Together, these commands allow you to control the probe station from your host computer.

The symbol preceding a command identifies its instruction set:

- \$: precedes a meta command
- : precedes a probe-station control command
- * precedes a GPIB command

Instrument vs. Controller

Cascade probe stations have both GPIB *talker/listener* and *controller* capabilities. By default, a probe station is a GPIB instrument. It becomes a GPIB controller when directed to make a VNA calibration, to run a computer program, or to set the temperature on a thermal controller. The probe station gives up control of the GPIB and reconnects to the host when it returns to

the local-operation mode or when it completes the calling of a program.

After a probe station has been in controller mode, the host must issue an IFC (Interface Clear) command to regain control of the probe station. This is because the GPIB switch box breaks off communication with the host to allow the probe station to control instruments. When the connection with the host is re-established, the host must consider the GPIB to be in an unknown state and initialize the bus.

NOTE *If you plan to use your probe station both as an instrument and as a controller, you will need Cascade's optional GPIB switch box. For example, if you plan to use thermal equipment or VNA calibration software and control the probe station from a remote host.*

Command and Query Responses

Command-complete and query-response handshaking conforms to the IEEE 488.2 standard. However, using the meta command **\$:set:resp on**, you can enable the sending of return strings so that the probe station sends a response string to the host after each command execution.

When return responses are enabled, the probe station sends the host one of the following after each command or query:

- the ASCII string "COMPLETE"
- an error message
- a query response

Unlike commands, queries always return a response, regardless of the meta-command settings. This is because the host expects an ASCII string in response to a query. If the host receives no response, it can hang. If the probe station cannot return the

expected response, it returns an error. Examples of query responses include a set of coordinate values, a version string, and a probe-plan file name.

As an example, suppose you want to upload a probe plan from the probe station computer to the host. First, you open a file on the host to receive the information. However, when you issue the **:probeplan:upload** command, you use an incorrect pathname. Since the probe station cannot locate the probe-plan file, it uploads an error message into the specified host location instead. Your host destination file contains only two line:

```
@Unable to open ProbePlan disk file.  
END OF FILE
```

The @ (at sign) precedes all error messages sent by the probe station, and the character string `END OF FILE` terminates all file transfers.

Setting Up for Remote Communication

This section explains how to set up and verify GPIB communication between the probe station and a remote host. The main topics are:

- Setting up GPIB communication parameters
- Verifying GPIB communication
- GPIB communication problems

Setting GPIB Communication Parameters

When operating as an instrument, the probe station requires only one address assignment in the host GPIB program. When the probe station is functioning as a controller, it uses GPIB address 0.

If the remote host and the probe station specify different GPIB addresses, communication across the GPIB will not occur. Incorrect or inconsistent addressing is one of the most common causes of communication problems.

Follow this procedure to set up GPIB communication parameters on the probe station, then on your remote host. Instructions for setting up addressing and timeout values on your host are general only, and will differ depending on the GPIB software installed on your host.

On the probe-station computer:

GPIB Server software requires installed Command Server software.

1. Double click the GPIB icon in the Cascade program group. The probe station launches both the GPIB and Command Servers. The screen displays the GPIB Server window shown in Figure 3-2.

NOTE: *The Command Server has no Windows menu interface. It's operation is transparent.*

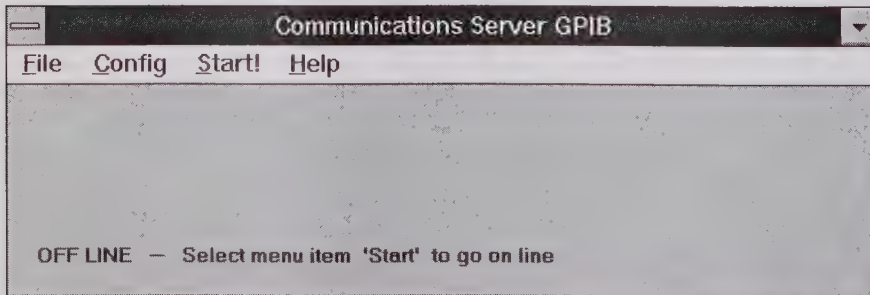


Figure 3-2. GPIB Server window

2. From the Config menu, choose **GPIB setup** to open the dialog box shown in Figure 3-3.

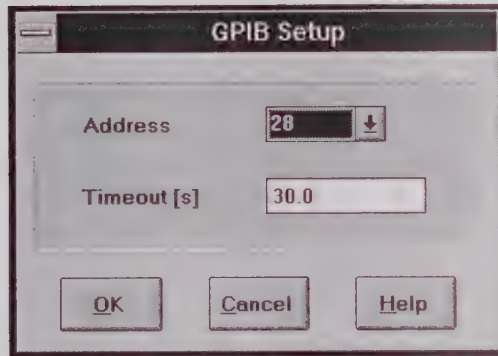


Figure 3-3. GPIB Setup dialog box

You will find more information later in this chapter under *GPIB Timeout Values*.

During your remote session, you can always clear the status field by choosing **Clear status now** from the Config menu.

3. In the ADDRESS field, enter the probe station's GPIB address. This address must be identical to the address assigned to the probe station by the host. The GPIB address for probe stations is usually 28.
4. In the TIMEOUT field, enter the time (for example, 30.0 seconds) you want the probe station to wait for a response from the host.
5. Click **OK**.
6. From the Config menu, select either **Pause on error** or **Terminate on error**. If an error occurs, pausing causes the GPIB Server to display the error message and wait for you to press **Enter** before executing the next command. Terminating ends the remote session if an error occurs. If you don't make either selection, the GPIB Server ignores errors.
7. (Optional) If your remote application program uses commands left over from Cascade's MS-DOS the probe station Software, select **Expect DOS 10K syntax** from the Config menu. This allows the GPIB Server to translate MS-DOS Summit commands to the Windows version syntax and to ignore obsolete commands. Otherwise, the server treats all MS-DOS Summit 10000 commands as errors.

8. (Optional) If you want to clear the status field every time a new input string is received, select **Clear status upon new input** from the Config menu.
9. (Optional) If you want to clear the response field every time a new input string is received, select **Clear response upon new input** from the Config menu.

Once you have set the probe-station parameters, you are ready to set the GPIB parameters on your host.

On your host:

1. Assign the probe station the same address in the host GPIB map as is defined on the station. This address is usually 28.
2. Set the host timeout value for the probe station to an appropriate length of time (for example, 30.0 seconds).
3. Set other communication fields as appropriate.

Once communication parameters are defined, you are ready to verify communication between the probe station and remote host.

Verifying GPIB Communication

If you are using a National Instruments GPIB interface card and software, you should also have the Interface Bus Interactive Control (IBIC) program. The IBIC program allows you to enter *send* commands to GPIB instruments from the host keyboard.

If you are not using a National Instruments GPIB, refer to your GPIB manufacturer's literature for information about sending commands directly from the keyboard.

The command-line examples in this guide assume you are entering remote commands directly from the keyboard.

Follow these steps to verify GPIB communication. If you have problems communicating with the probe station, refer to the troubleshooting information following this verification.

1. Make sure the GPIB communication parameters on the probe station are set appropriately, as described earlier in this chapter.
2. Bring the probe station online for GPIB communication by clicking **Start!** in the GPIB Server window. The window changes to display GPIB status-reporting fields, as shown in Figure 3-4.

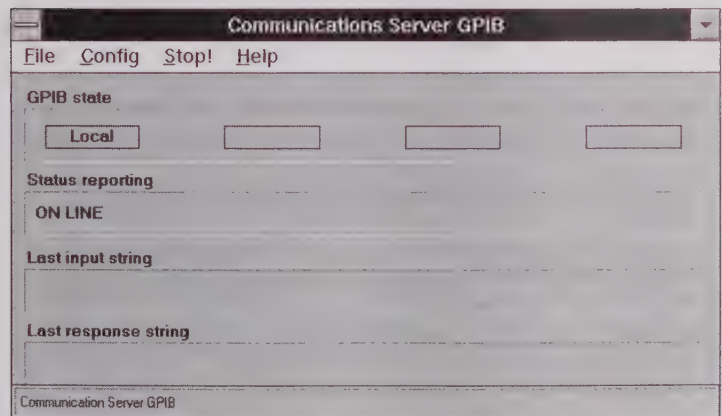


Figure 3-4. GPIB Server window when server is online.

The host can sense when the probe station is ready to receive commands by serially polling the station and checking for bit 2 to be set to 1. Serial polling returns the status byte, and bit 2 is the third bit (0000 0100) since the first bit is bit 0. When the probe station leaves remote mode, bit 2 returns to 0.

3. Make sure your host is set up as a GPIB controller.
4. Make sure the address specified on your remote host is identical to the address specified on the probe station.

Entering
commands in
this method
uses string-
return
handshaking.

5. Make sure the timeout value is appropriate for your application.
6. On your remote host, invoke the National Instruments IBIC program (or an equivalent) so that you can enter commands directly from the host keyboard. In these examples, the command prompt is > (right caret).
7. Enter the following identification command. The probe station should respond with an ID string, which will differ depending on your model number.

```
> *idn?  
Cascade Microtech, Probe Station, 0,  
GPIB Server 1.00
```

You have now made sure remote-communication parameters are set properly and that the probe station returns an expected value.

This next procedure shows how to verify stage-movement functionality and interface features.

1. Make sure you can read status registers by sending the following status-byte (STB) register read command. The value returned will differ depending on the value stored in your STB register:
2. Display a one-line message on the probe-station screen. The following command line shows an example message sent with the abbreviated form of the **:system:display** command.
3. Now use a stage-movement command to move the stage to the load-device position, at the front of the station. (The load-device command is described in detail in *Section 4* of this guide.)

```
> *stb?  
19
```

```
> :syst:disp "Communication verified."
```

```
or  
> :move:load 1 (for Summit 10000-series)  
> :move:load 2 (for Summit 12000-series)
```

If you receive appropriate responses and see the stage move to the load-device position, communication across the GPIB is verified. If you have problems communicating with the probe station, refer to the next discussion.

Identifying GPIB Communication Problems

Especially when lab equipment is shared between people, communication addresses and connections can be unexpectedly changed. One of the most common communication problems is that cables are not plugged in. Another common problem is that the remote instrument is not online for remote communication. The following list briefly describes common problems that prevent instrument communication.

- The bus cable is not plugged in to both components.
- The bus cable is plugged into the wrong component.
- The address assigned to the probe station is incorrect.
- The address assigned to the probe station is already assigned to another instrument.
- Power to the probe station is not on.
- The host has not taken remote control of the probe station.
- The probe station is not specified on your host as being online for GPIB communication.
- You did not click **Start!** in the GPIB window to enable remote mode. If remote mode is not enabled, the probe station cannot listen to or talk with the host.
- You disabled remote communication by clicking **Stop!** in the GPIB window.
- Probe-station timeout values are not long enough to allow the probe station to respond to requests and commands from the bus.
- The host timeout values assigned to the probe station are too short for the host to communicate with the station.

- For the GPIB card installed in your host computer, check the GPIB port address and interrupt levels set into the bit switches on the GPIB card. These bit switches must match the port address and interrupt levels defined in the host GPIB software.

Check all items previously listed before assuming that your hardware or software is not operating properly. If you still cannot communicate with your instrument, call the manufacturer of your GPIB card for service.

Handshaking, Status Reporting, and Time-outs

The method of handshaking used for remote communication must be appropriate for the instruments connected to the bus. In the case of a probe station, the handshaking method affects the ability of the host to correctly determine whether the probe station has completely finished executing a remote command. This section explains the differences between methods of handshaking, and makes suggestions for selecting an appropriate method for remote communication with a probe station. To help you understand handshaking, this section also explains the GPIB status-reporting structure and discusses timeout values.

Cascade probe stations support three methods of handshaking for remote communication between the probe station and the host. Each of these methods is discussed in detail later in this section.

- Serial poll
- Service request (SRQ)
- String-return

This section uses excerpts from a program to illustrate remote-communication concepts. The following list summarizes the conventions used:

- Bits in a byte are numbered 0 through 7, with 0 being the least significant bit.
- The probe station uses positive logic: true = 1; false = 0.
- The variable *device* represents the GPIB address of the probe station.

Status-Reporting Structure

Cascade's probe-station status-reporting structure is shown in Figure 3-5. Refer to this figure throughout this discussion to see how bit settings are masked and passed from one register to another.

Instrument-Defined Bits

IEEE 488.1 and 488.2 provide several instrument-defined bits in the status-byte (STB) register. Cascade probe stations define bit 0 of the STB register to be an error (ERR) flag. If a command, query, or execution error occurs, the probe station sets the appropriate bit or bits in the event-status (ESR) register. The probe station then copies a true value (1) into the ERR bit of the STB register.

Bit 2 of the STB register is the online (ONL) bit. This bit is set to 1 when the GPIB Server comes online and to 0 when it goes off-line. In other words, the ONL bit is set when you click **Start!** in the GPIB Server window and cleared when you click **Stop!**. The host can sense when the probe station is ready to receive commands by serially polling the probe station and checking for bit 2 of the STB register to be set to 1. When the probe station leaves remote mode, bit 2 returns to 0.

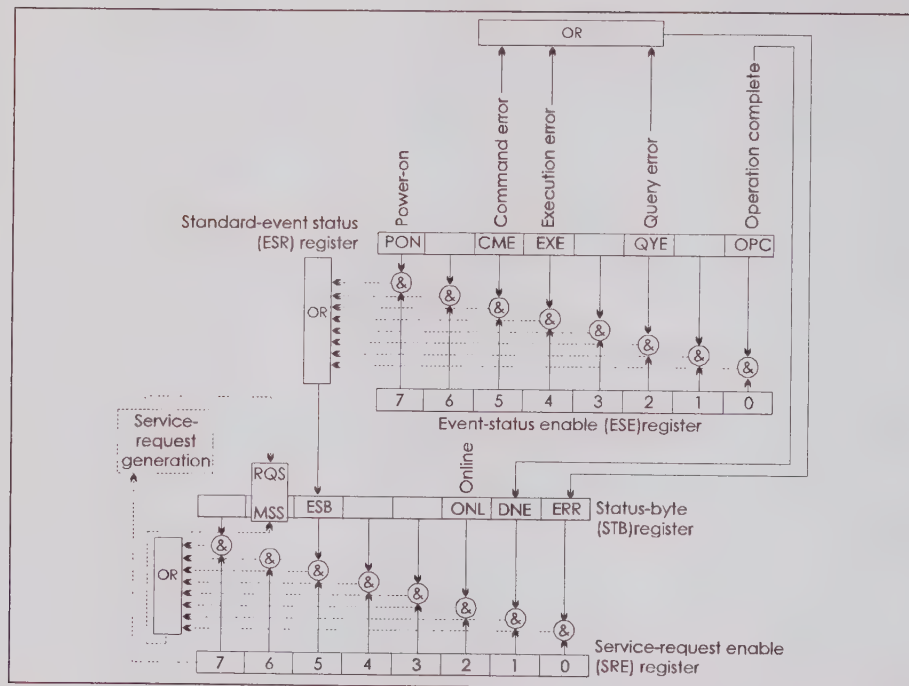


Figure 3-5. Status-reporting structure

Figure 3-5 also shows that the probe station defines bit 1 of the STB register to be a done-bit (DNE). This bit corresponds to the OPC bit in the ESR register. When the probe station sets the OPC bit to 1, the DNE bit is also set to 1. This allows the host to use a serial poll to quickly determine if operations have completed successfully.

For example, you might not be able to begin a measurement until the stage is positioned properly. In your host program, you can use the GPIB ***opc** command immediately after the **:move:absolute** command. You can then tell the host to wait until the OPC bit is true before continuing to the next program instruction.

Masking

As you can see in Figure 3-5, the event-status enable (ESE) register allows you to mask specific events that occur in the ESR register. In this discussion, for simplicity, the value of the ESE register is 1. This means that all bits except bit 0 (which corresponds to the OPC bit) are masked. In this case, only an operation-complete flag can filter through the status-reporting structure and generate a service request.

For example, the host program sends a command string, with the last command being the ***opc** command. When the probe station completes the operations specified by the command string, it sets the OPC bit in the ESR register. When the OPC bit is set, that value is ANDed with the value of bit 0 in the ESE register. Because at least one event is true, the event-status bit (ESB) in the STB register is then set.

There are two registers you can use to mask events from causing service requests. The ESE register masks events that occur in the event-status register. This stops events from setting the event-status bit in the STB register. However, this does not prevent errors or operation-complete flags from being set in other bits of

the STB register. For example, the ESR register OPC bit setting is copied into the DNE bit. Therefore, the STB register bit 1 can still be set true, regardless of the ESE register value.

The SRE register allows you to mask events in the status-byte register. For example, if a power-on state is reported and filters down to set the ESB bit true, you can set bit 5 of the SRE register to 0. This masks the power-on state, preventing it from causing a service request.

Reading the STB Register

You can read the STB register using two methods. You can send a GPIB ***stb?** query, or you can set up the host to serial-poll the STB register. Serial polling is described later.

Bit 6 of the STB register is the master-status/request-for-service (MSS/RQS) bit. The meaning of the value of this bit depends on the command used to read the register. If you read the STB register with ***stb?**, a true value in the MSS bit indicates that a service request is pending or has occurred. When you use the ***stb?** command, the host cannot tell if the service request has actually been generated. However, if you use a serial poll to read the STB register, a true value in the MSS bit indicates that a service request has actually been generated.

Clearing the Registers

Use the GPIB ***cls** command to clear most of the registers and reset them to 0. The ESR register cannot be cleared by ***cls**. Refer to the *GPIB Command Summary* at the end of this chapter for specific information about the registers affected by the ***cls** command.

NOTE: Before you send a meta or probe-station command, you must send a GPIB ***cls** command to clear old bit settings from the status registers. If you are using a serial poll without SRQs, you must send ***cls** as a separate string.

GPIB Timeout Values

One of the most common communication problems is a timeout value that is too short to allow the instrument to complete the task. You need to choose a timeout value that is long enough for any task your application might perform. For example, if the longest task is to slowly move the stage from one location to another, a timeout value of 30 seconds may be sufficient. However, if your program requires the operator to perform a task such as aligning theta or loading a wafer, you may want to increase the time to 5 minutes or more.

The timeout value becomes particularly important if your program sends messages without checking the status to see if the operation completed or without reading the response string for verification. In many applications, relying on timeout values is not as effective as using SRQs or serial polling to synchronize the host and the probe station (see *Alternative Timeout Methods* later).

What the Timeout Value Does

The timeout value on the probe station indicates how long the probe station will wait for the host to read a response sent by the probe station.

The timeout value used in the host program controls how long the host waits for the probe station to respond after the host sends a command. Once the host has issued a command, the host should be ready to read the ASCII string returned by the probe station.

If the remote task completes within the time allowed, the host reads the returned string, and is ready to execute another task.

If there was no timeout value for the probe station, the host would wait indefinitely for a response if the probe station could not finish the host-specified task,. This, in effect, hangs the host. The timeout value tells the host that, after a certain period of time, it should stop waiting for the probe station to respond, and should then report an error, perform some other task, or be ready to perform some other task.

What Happens During a Timeout

Cascade probe stations treat a timeout the same as any other error. So, the selections you make in the Config menu determine the probe station's behavior. If you check **Pause on error**, the GPIB Server displays a timeout message and waits the operator to press **Enter** before accepting additional commands from the host. If you check **Terminate on error**, the GPIB Server application terminates whenever a timeout occurs. If you don't check either of these Config options, the GPIB Server displays the timeout condition in the **Status reporting** field and continues to receive new commands. It is possible, then, that no one would notice that a timeout occurred, which could result in the probe station being out of synchronization with the rest of the commands in your program.

The most important consideration in setting the timeout on the probe station is that it is the maximum time the host has to pick up a response from the probe station.

NOTE *The GPIB Server times out if it does not receive responses from the Command Server as well as from the remote host.*

How you configure your host to handle a timeout will probably have a greater impact on your application than the probe station

since the probe station, by default, keeps listening for commands even after the timeout time has been exceeded.

Setting the Timeout Value

On Cascade probe stations, the timeout value is set in the GPIB Setup dialog box. Access this by choosing **GPIB setup** from the Config menu in the GPIB Server window.

On the host, the probe-station timeout value is set in your host GPIB program.

Picking an Appropriate Timeout Value

The timeout value you set must be long enough to allow you to complete the longest task you expect to begin. For example, if you expect only to remotely move the stage from one location to another at a slow speed, a timeout value of 1 minute might be appropriate. But, if you will be performing a manual operation such as **:probeplan:align**, you might want a timeout value of 10 minutes or more. Or, consider using a serial poll or SRQ handshaking to determine when the command execution is complete without any risk of a timeout.

As a general rule, you can expect the following types of commands to take longer than a few seconds to execute:

- :move:scan commands
- :move:load
- :set:acceleration
- :set>manual
- :set:velocity
- :system:delay
- :system:display

For a National Instruments GPIB, you can increase the timeout value to approximately 10 minutes. You can disable the timeout by setting the value to 0. However, if you disable the timeout value and the probe station cannot complete a remote command, the host can hang, waiting indefinitely for the probe station to respond.

Alternative Timeout Methods

Probe stations let you perform many tasks that can take a relatively long time to complete. For these tasks, consider using one of the following methods instead of relying on a timeout value:

1. You can use a serial poll and/or SRQs with your own timeout handling or keyboard-check handling. See *Using Service Requests* later in this chapter.
2. You can disable or set a very long timeout value before sending the command to the probe station. After the command is completely executed and the host has received the probe station "COMPLETE" message, you can reset the timeout value.
3. You can cancel remote mode, perform the task, then invoke remote mode again.
4. You can write your own loop on the host so that a timeout is not considered an error.

Using Serial Polls

The serial-poll method of handshaking is simpler and faster than other methods of communicating across the GPIB. This method is illustrated in the routine shown in Figure 3-6.

In Figure 3-6, the *ReadStatusByte* procedure is from the National Instruments 488.2 software interface to the National Instruments

GPIB board. The *ReadStatusByte* procedure makes a serial poll. The 488.1 serial poll will also work.

To illustrate how serial polling works, let's follow an operation-complete signal through the status-reporting structure. Figure 3-6 shows that the ***cls** instruction is sent before each probe-station command. This instruction clears most of the registers in the status-reporting structure.

Important: The ***cls** command must be sent in a separate string, *before* the string of operation commands is sent. However, the ***opc** command can be the last command in the operation-command string. This avoids timing problems in case the host starts a serial poll before the STB register has been cleared.


```
.  
.  
{===== Poll_Until_Done =====}  
PROCEDURE Poll_Until_Done  
  ( device : INTEGER ) ;  
  
CONST  
  DONE_BIT = $02 ;           { bit 1 is hex 2 }  
VAR  
  status : INTEGER ;  
BEGIN  
  Writeln ( 'Polling status until complete bit s.' ) ;  
  { Read the probe station status byte until the OPC bit is set }  
  REPEAT  
    ReadStatusByte ( board_0 ,  
                      device ,  
                      status ) ;  
  UNTIL ( status AND DONE_BIT ) <> 0 ;  
END ;                          { procedure Poll_Until_Done }  
  
.  
.  
.  
{ The main program is now executing, and the stage is  
  positioned at the first die site marked for testing.  
  The next command raises the chuck. Note that you must  
  clear the status-reporting structure before sending a  
  command, so that the OPC bit setting does not reflect a  
  previous command, but is ready for the new command's  
  completion. }  
Send_Cmd ( device , ' *CLS ' ) ;  
Send_Cmd ( device , ' :MOVE:CONTACT 1 ; *opc ; ' ) ;  
Poll_Until_Done ( device ) ;  
  
.  
.  
.
```

Figure 3-6. Sample Poll_Until_Done routine

After each command is sent, the program issues an ***opc** instruction. The ***opc** instruction sets the OPC bit when the previous command (or set of commands) is finished executing.

In the example, when the program calls the Send_Cmd routine, the host sends a command string to the probe station. The last command in the string is ***opc**. As soon as the string has been sent, the host begins executing the next instruction in the program. Because the next program instruction is the Poll_Until_Done

routine, the host begins polling the status byte to see if the DNE bit is set.

In Figure 3-6, bit 1 of the STB register is identified by the hexadecimal value 2 (\$02).

When the last command in the previously sent command string is completed, the probe station has executed the ***opc** instruction. When the ***opc** instruction executes, the OPC bit is set to true. This setting is copied to the DNE bit. When the DNE bit is true, the serial poll returns that information to the host. The host then assumes that the previously sent operation (the command string) has completed successfully and the probe station is ready for the next program instruction.

The routine shown in Figure 3-6 is simplified for this example. In an actual application program, consider adding lines that allow you to terminate the program by pressing a key on the probe-station keyboard. This is especially useful if an error occurs or communication hangs.

Using Service Requests (SRQs)

The GPIB status-reporting structure allows you to use and control service requests (SRQs). In this method of handshaking, the host can interpret an interrupt as a request for attention from a particular device. To control service requests, you use the event-status and service-request enable registers. These registers are shown earlier in Figure 3-5. Figure 3-5 also shows the defined bits (bits 0 and 1) in the STB register.

As an illustration of how service requests might be used, examine the code fragment listed in Figure 3-7. This program excerpt calls three routines: `Turn_On_SRQ`, `Wait_For_SRQ`, and `Turn_Off_SRQ`.

In Figure 3-5, you can see that the ESE register allows you to mask specific events that occur in the ESR register. In this discussion, for simplicity, the value of the ESE register is 1. This

means that all bits except bit 0 (which corresponds to the OPC bit) are masked. So, only an operation-complete flag can filter through the status-reporting structure and generate a service request.

```

        .
        .
        .
    { Instruct the probe station to generate an SRQ when done
with
    commands }
Turn_On_SRQ ( device ) ;
    { Instruct the probe station to move to the center position
of
    the stage. First clear the registers and then ask for
    the OPC bit to be set when done (home is found). Since
    SRQ now is enabled for this event, we will simply wait
    for an SRQ afterwards. Other processing could have
    taken place, if so desired. }
Send_Cmd ( device, '*CLS; :MOVE:CENTER 1; *OPC;' ) ;
Wait_For_SRQ ( device ) ;
    { Load a ProbePlan (die site map) from the probe station mass
storage into the probe station memory. This command will
also
    require the operator to joystick the probes over the
    alignment die, then move to the first die. The Probe-
    Plan can also be loaded locally via the probe station
inter-
face. Clear the registers and wait for SRQ again. }
Send_Cmd ( device, '*CLS; :PROBEPLAN:LOAD 1 PP001.PPD; *OPC;' )
;
Wait_For_SRQ ( device ) ;
    { Instruct the probe station to stop generating an SRQ when
done
    with commands. Not needed for the following "define
    location" commands. }
Turn_Off_SRQ ( device ) ;
        .
        .
        .

```

Figure 3-7. Turning on, waiting for, and turning off SRQs

When the probe station completes the specified command string, it sets the OPC bit in the ESR register. When the OPC bit is set, that value is ANDed with the value of bit 0 in the ESE register. Because at least one event is true, the event-status bit (ESB) in the STB register is then set.

At this point, there is a value in bit 5 (the ESB bit) of the STB register. Because the probe station also copies the value of the OPC bit to the DNE bit, STB register bit 2 also has a true value. The register should now have the value 34. For an SRQ to be generated now, the other mask register (SRE) must have a 1 value in the bit which corresponds to the STB register's ESB bit. The setting of the ESB bit, which is the summary of enabled bits in the ESR register, then generates an SRQ.

There are two methods you can use to read the STB register. You can send a ***stb?** query, or you can set up the host to serial poll the STB register. Serial polling is described earlier in this chapter.

When the ESB bit is set to 1, it indicates that at least one enabled condition of ESR has occurred. For example, if the SRE register bit 5 is set to 1, a service request is generated. Bit 6, the MSS/RQS (master status/request for service) bit is then set to 1, indicating that the service request has occurred.

Refer to the serial-poll discussion for information about a more efficient method of handshaking.

In this example, the operation-complete flag filters through the status-reporting structure until it generates an SRQ. As you can see in Figure 3-7, the host, seeing the SRQ, stops waiting for a response and goes on to the next remote command.

In general, you should set up your status-reporting structure so that only the ESB bit in the STB register can generate a service request. Do not set up your status-reporting structure so that other STB bits generate an SRQ. You should control SRQ generation with the SRE register bit, which corresponds to the STB register's ESB bit.

The probe station allows you to perform tasks which can take an indefinite amount of time. For example, adjusting theta. Because of this, you must design your host program so that it waits for the SRQ, regardless of how long it might take. In general, you should avoid using the timeout value in the wait-for-SRQ loop. Instead, substitute a keyboard-check routine. For example, you

might create a loop that accepts **Esc** (on the host keyboard) as an abort key for halting the remote task and returning control to the host.

Using String-Return Handshaking

The string-return method of handshaking is the simplest form of remote communication. When using this form of handshaking, you send a command to the specified instrument and wait for a response. After you receive an appropriate response, you send the next command.

NOTE *String-return handshaking does not follow IEEE 488.2 rules. It is supported to provide backward compatibility with existing application programs.*

Once you invoke your host GPIB program, you can send individual commands one at a time across the GPIB. Depending on the setting of the commands, the probe station responds differently when an operation is complete. For string-return handshaking, the probe-station commands should have the settings shown in Table 3-1.

The settings recommended in Table 3-1 allow the probe station to return an ASCII string after each command is successfully completed (or an error occurs). After each command is successfully executed, the probe station returns the ASCII string "COMPLETE". If an error occurs, the probe station returns the ASCII error message instead. Also, if the command was a query, the probe station returns the response only (or the error message only). The first character in an error message is always the @ ("at") sign.

Table 3-1. Recommended Meta-Command Settings^a

COMMAND	RECOMMENDED SETTING
\$.set:show:command	on/off
\$.set:show:pause	off
\$.set:respond	on
\$.set:error:abort	on/off
\$.set:error:pause	off

^a for string-return handshaking only

For GPIB communication, the string-return method of handshaking relies on a timeout value. The timeout value on the host indicates to the host how long the host should wait for a response from the probe station.

If your host uses the National Instruments GPIB board, your manufacturer's documentation describes the IBIC program. IBIC provides you with the string-return method of handshaking for GPIB communication. You can also use IBIC to test serial-poll and SRQ handling.

Understanding Command Execution

Refer to the timeout discussion earlier in this chapter.

Cascade probe stations execute commands in sequence as they arrive on the bus. With the exception of a few move commands, a probe station does not read a new command from the bus until it has finished executing the last command. This means that you must make sure your timeout value is long enough to allow the probe station to complete all tasks, or you can use an alternative timeout strategy.

Commands and queries send different responses to the host, depending on the settings of the meta commands and whether or not an error occurs.

The GPIB commands (described in detail later) allow you to specify the response the probe station sends the host after a remote command has executed. You can specify that the ASCII string "COMPLETE" or an error is sent. You can also specify that the probe station does not return a response to the host after completing a command.

Queries return a response, regardless of the settings of the meta commands. This is because the host expects an ASCII string in response. If the probe station cannot return the expected query response (such as a set of coordinate values, a version string, or a ProbePlan file), the probe station returns an error.

The "END
OF FILE"
string
terminates all
file transfers.

The @
(at-sign)
precedes all
probe-station
error
messages.

For example, you might want to upload a ProbePlan from the probe station to the host. So, you open a file on the host in which to store the uploaded information. However, when you enter the upload command, you incorrectly specify the ProbePlan path. Because of this, the probe station cannot locate the ProbePlan file. Since the ProbePlan file cannot be located, the probe station uploads into the specified host location the error message instead. In this example, if you have opened a file on your host in which to store the ProbePlan contents, the host file contains only the following two lines:

```
@Unable to open ProbePlan disk file.  
END OF FILE
```

It is up to the host program to specify a location for the up-loaded information.

Types of Commands

Cascade probe stations provide you with several types of commands for remote communication:

- Probe-station control commands
- Meta commands
- GPIB protocol commands

NOTE *In previous versions of Cascade probe-station software, alternate command sets (alt1 and alt2) provided additional techniques for controlling probe stations. These command sets will generate syntax errors if you attempt to use them with this version of the GPIB Server.*

Make sure you read about command execution and timeout values before using remote communication across a GPIB.

Probe Station Commands

Probe station commands control the probe station. For example, they move the stage, turn vacuum on and off, load probe plans, and so on. They use the same syntax as Command Server commands. Chapter 4 of this guide contains a description of this command syntax.

Meta Commands

Table 3-2 describes the Meta commands.

Meta commands control system level-functions such as whether the computer displays commands and responses. Meta commands are preceded by a \$ (dollar sign), but have SCPI-compatible syntax (similar to probe-station commands). Meta commands comply with these general rules:

- The commands are not case-sensitive.
- Meta commands always start with a \$ (dollar sign).
- Parameters to a command follow the command and are separated by spaces.
- There is also a space between the command and its first parameter.
- Commands are separated by a ; (semicolon).
- The keyword separator for multiple keyword commands is : (colon).
- Meta commands do not return a response.
- All commands and responses are ASCII strings.

For example, you can use the meta command sequence **\$:set:respond on**. This command specifies that the probe station returns to the host the message "COMPLETE" after each successfully completed remote operation.

Table 3-2. Remote-Mode Meta Commands^a

COMMAND	DESCRIPTION
\$:set :err :abor [on] [off]	Terminates the GPIB Server session and returns the probe station to local control when an error occurs. This is the same as selecting Terminate on error in the Config menu.
\$:set :err :paus [on] [off]	Waits for the user to confirm the error message by pressing a key on the probe-station keyboard before continuing to the next command. This is the same as selecting Pause on error in the Config menu.
\$:set :mode [native] [alt1] [alt2]	This is an obsolete command that the GPIB Server ignores. The GPIB Server recognizes :set:mode native to provide backward compatibility with existing application programs. Using alt1 or alt2 generates an error.
\$:set:repeat	When you're reading a file instead of the bus, this command causes the GPIB Server to reopen the file and read it again. In general, you would use this only for debugging purposes. To terminate the repeat loop, click STOP! in the GPIB Server window.

(continued on next page)

Table 3-2. Remote-Mode Meta Commands- *continued*^a

COMMAND	DESCRIPTION
\$:set :resp [off] [on]	Sends an ASCII string "COMPLETE" to the host after each operation is completed unless an error occurs. If an error occurs, uploads the error message to the host. NOTE: This is nonstandard for GPIB communication, which usually polls the status byte or uses serial polling.
\$:set :show :comm [off] [on]	Displays each command as it is received (not executed) from the remote host and each response as it is sent to the host. The GPIB Server window expands to provide display space.
\$:set :show :resp [off] [on]	This command is ignored. The GPIB Server recognizes it only to provide compatibility with older MS DOS-based application programs. Use :set:show:comm instead.
\$:set :show :paus [on] [off]	Waits for you to confirm the display with a keystroke from the probe station keyboard before continuing to the next command.
\$:set:syntax [win] [dos]	Determines whether the GPIB Server will recognize only commands supported in the Windows version or will recognize obsolete commands from the MS DOS version. The GPIB Server defaults to Win (dows), which treats obsolete commands as errors. Sending the dos parameter allows the GPIB Server to interpret or ignore obsolete commands. This is the same as selecting Expect DOS 10k syntax in the Config menu.

^a When parameters are stacked, choose one.

GPIB Protocol Commands

The *Status-Reporting Structure* section of this chapter also discusses GPIB commands.

GPIB protocol commands read, set, and clear GPIB status-reporting registers. The GPIB Server uses the GPIB protocol commands and does not pass them to the Command Server. Table 3-3 describes the GPIB commands.

GPIB commands are preceded by an * (asterisk). For example, if you wanted to set the service-request enable register (SRE) to a particular value, you would use the ***sre** command.

Queries, which return a response to the remote controller, are followed immediately by a ? (question mark). For example, if you wanted to know the status of the SRE register, use the ***sre?** query command.

Table 3-3. GPIB Protocol Commands

COMMAND	DESCRIPTION	FUNCTION
*cls	clear-status	Clear-status command. Clears the ESR, and STB registers in the status-reporting structure. This command forces the probe station into the operation-complete command-idle state and the operation-complete query-idle state. This command does not clear the ESE and SRE registers.
*ese	event-status enable	Sets the value of the ESE register. You can use *ese to reset this register to all 0s.
*ese?	event-status enable	Returns the value of the ESE register.

(continued on next page)

Table 3-3. GPIB Commands- continued

COMMAND	DESCRIPTION	FUNCTION
*esr?	event-status register	Returns the value of and clears the standard event-status register (ESR).
*idn?	identification query	Returns the probe station identification string.
*opc	operation-complete	Sets the OPC bit in the ESR register and bit 1 (the ERR bit) in the STB register.
*opc?	operation complete	Places a 1 in the output queue when all query selected pending operations are completed.
*rst	reset	Resets the probe station. This command is the same as the *cls command.
*sre	service-request enable	Sets the value of the SRE register (the mask register for SRQ generation). For example, you can use this command to reset this register to all 0s. This is an immediate command.
*sre?	service-request enable query	Returns the value of the SRE register.
*stb?	status byte query	Returns the value of the STB register.
*tst?	self-test query	Always returns a 0.

For more information about GPIB commands, refer to the IEEE 488.1-1987 and 488.2-1987 documents. The use of GPIB commands in the Cascade probe-station status-reporting structure is described earlier in this guide.

Comments and Messages in Programs

If you want to include comments in your program, precede them with double slashes (*//*). For example:

```
//This is a comment.
```

Comments can be in any position in the program file.

The probe station displays messages in a Windows message box on your probe-station monitor. If you want to include messages in your program, precede them with the at sign (*@*). For example:

```
@This is a message.
```

Advanced Troubleshooting

The *C:\CASCADE\CSGPIB.INI* file contains the initialization information for the GPIB Server. This file is created the first time you exit the GPIB Server (so it will not exist if you have never run or properly exited the GPIB Server). The GPIB Server updates the *CSGPIB.INI* file whenever you exit the server. In this way, the GPIB Server always comes up with the settings that were in place the last time you ran it. If you need to, you can delete this file because it will be recreated the next time you exit the GPIB Server.

Figure 3-8 lists a modified *CSGPIB.INI* file. You'll notice that this file contains three sections: Debug, Timing, and Options. The file that is created when you exit the GPIB Server contains only the Options section. The Debug and Timing sections were manually added, as will be explained a little later.

The Options section of the *CSGPIB.INI* file records the selections made in the GPIB Server Config menu. In this file, **1** corresponds to *on* and **0** corresponds to *off*. So, in Figure 3-8, the user has selected **Pause on error** in the Config menu rather than **Terminate on error**.

```
[Debug]
logFile=0
inputFromFile=1
inputFileName=gplibmove.gpb

[Timing]
gpibTimerMs=100

[Options]
pauseOnErr=1
terminateOnErr=0
isDos10KSyntax=0
dispInputOutput=1
talkListenAdr=28
hostAdr=0
targetAdr=16
autoClearResponse=0
autoClearStatus=0
```

Figure 3-8. CSGPIB.INI file

To assist you in troubleshooting GPIB communication problems, you may want to add a Debug or Timing section to the *CSGPIB.INI* file. If you do this, you must use the same keywords as shown in Figure 3-8.

The keywords in the Debug section are as follows:

Keyword	Does This
logFile	When set to 1 (turned on), causes the GPIB Server to record events in the <i>C:\CASCADE\CSGPIB_D.LOG</i> file. Recorded events include commands, responses, interpretations of DOS syntax commands, and other useful debugging information. Set logFile=0 to disable logging.
inputFromFile	When set to 1 (turned on), causes the GPIB Server to read a file rather than the input from the bus. The inputFileName keyword specifies the file that will be read. Set inputFromFile=0 to read from the bus.
InputFileName	Specifies which file will be read when inputFromFile is turned on.

Notice that in Figure 3-8, the GPIB Server is set to read from the *GPIBMOVE.GPB* file rather than the bus. Figure 3-9 lists this file. Basically, this file sets a position and moves the stage a few times. Let's step through this file line-by-line so you'll know how to create a file in case you ever want to do this for troubleshooting your application program:

- `$:set:syntax windows` is a meta command that essentially tells the GPIB Server to treat obsolete commands as errors. Meta commands begin with a \$.

- Double slashes (//) at the beginning of the next line identify it as a comment that the GPIB Server will ignore. Comments can begin in any column, but cannot wrap.
- The at sign (@) at the beginning of the third line makes it a message that will be displayed in a message box.
- The lines that begin with a colon (:) are commands that control the probe station. You'll find descriptions of these commands in Chapter 4 of this guide.
- Immediately following probe-station commands that are queries, is a line containing a single question mark (?). This simulates the host reading a response.

```
$:set:syntax windows
// This is a comment
@ This message will be displayed in a MessageBox
:set:preset 2 0 0 0;
:move:abs 2 2000 2000 UP;
:move:rel 2 2000 2000 UP;
:move:abs 2 -2000 -2000 UP;
// Ask position and get answer
:move:abs?
?
:move:abs:nchuck 2 -2000 -2000 NONE;
```

Figure 3-9. GPIBMOVE.GPB file

The Timing section of the *CSGPIB.INI* file consists of a single keyword: `gpibTimerMs`. This sets the interval at which the GPIB Server polls the bus for input. The units for this keyword are milliseconds. So, the file shown in Figure 3-8 sets the GPIB Server to poll every 100 milliseconds. Sometimes, it is useful in troubleshooting to slow down the bus polling rate.

As stated earlier, the *CSGPIB.INI* file contains only the Options section by default. Adding the Debug and Timing sections is optional to help you troubleshoot GPIB communication problems.

4

COMMAND DICTIONARY

Introduction

This chapter contains an overview of the syntax for probe-station control commands, a list of the commands, and detailed descriptions of each command listed in alphabetical order. Most of these commands can be used in application programs that communicate with the DDE Server, the GPIB Server, or the Command Server. However, a few of these commands are only for use with the GPIB because they only have relevance in a remote control environment.

Some commands require a parameter which includes a unit of measurement. For these commands, the units are not specifically identified in the command summary. Instead the units (such as centimeters or inches; seconds or milliseconds) are identified in the command description. Use the **:set:unit** command to specify whether you will be using metric or English units.

Command Syntax

This guide uses the following conventions to describe command syntax:

- Boldface indicates a literal command or parameter which must be entered exactly as shown. For example, the **:move:load 2** command.
- Italics indicate a replaceable parameter for which you supply a value. Numerical values, file names, labels, and indexes are examples of replaceable parameters. For example, *pp001.ppd* might be the name of a probe plan file you want to upload to the host.
- Brackets [] indicate that there is a list of choices from which you must make a selection. You must choose one of the bracketed parameters listed in the stack of available settings.

The Command Server, the DDE Server, and the GPIB Server use the same command, which complies with SCPI syntax guidelines.

This list briefly describes the command syntax:

- Commands are preceded by a : (colon). For example, **:move:center 2**.
- Command sequences have colons between the parts, but do not contain spaces. For example, the **:move:probeplan:prior:die 2** command sequence.
- Commands and parameters are delimited by a space. For example, the following command has two parameters, which represent a set of coordinates: **:move:abs 2 42 -1178 up**.
- Multiple commands on a single command line are delimited by a semicolon. For example, the following command line moves the stage to the load position, turns off the vacuum, and returns the stage coordinates:

```
> :move:load 2; :set:vac 2 off; :move:abs? 2
```

- Queries, which return a response to the remote controller, are followed immediately by a question mark; for example, **:set:delay?**.

You can use commands in two forms: the complete (long) form and the abbreviated form. If you use commands in the long form, you must enter the entire command name. For example, you can enter the **:move:absolute** command, but not in the form, **:move:absolu**.

You can use commands in the long or abbreviated form.

If you use commands in the abbreviated form, you must enter the first three or four characters of the command. If the fourth character of the command is a consonant, the abbreviated form of the command is four letters long. If the fourth character of the command is a vowel, the abbreviated form of the command is truncated to three letters. If the command is only four characters long, the command cannot be abbreviated. Some examples of abbreviations are shown in the following table.

Table 4-1. Abbreviated Commands

LONG FORM	ABBR. FORM
:move:absolute	:move:abs
:move:probeplan:next:subsite	:move:prob:next:subs
:probeplan:set:palette	:prob:set:pal
:set:display:messages	:set:disp:mess
:system:identification	:syst:iden

Summary of Commands

Table 4-2 contains a complete listing of the commands for controlling Cascade Summit-series probe stations. These commands can be sent via the DDE Server, the GPIB Server, or directly from your application program. See *Chapter 3, GPIB Communication* for additional commands having to do with communication across the General Purpose Interface Bus.

NOTE

Some of these commands are described as GPIB only. This is because these commands only have meaning in a remote control context.

Table 4-2 Prober Control Commands^a

COMMAND SYNTAX	DESCRIPTION
<pre> :move:abs devID x y z [none] [none] [up] [down] [none] </pre>	Lowers the chuck, moves the stage to the supplied x- and y-axis positions, then raises the chuck to the supplied z-axis position. Positions are relative to the currently defined zero position. Use none as the z-axis parameter when you don't want any change in chuck height. The query returns the x-, y-, and z-axis coordinates for the current stage location and chuck height.
<pre> :move:abs:chuc devID x y z [none] [none] [up] [down] [none] </pre>	This command is the same as the :move:abs command.

COMMAND SYNTAX	DESCRIPTION
:move:abs:nch <i>devID</i> <i>x</i> <i>y</i> [none] [none]	Moves the stage, without first lowering the chuck. Positions are relative to the currently defined zero position. Begins to execute the next command in the queue before the move is complete (the stage positioned at its final destination). Use the move:wait command to make the probe station wait until this stage movement is complete before executing the next command.
:move:cent <i>devID</i>	Moves the stage to the center (0,0) of the stage-movement area.
:move:cont <i>devID</i> :move:cont? <i>devID</i>	Raises the chuck to contact the probes. The query returns TRUE if the chuck is up and FALSE if the chuck is down.
:move:down <i>devID</i> :move:down? <i>devID</i>	Lowers the chuck. The query returns TRUE if the chuck is down and FALSE if the chuck is up.
:move:home <i>devID</i>	Initializes the hardware at the center (machine 0,0) by moving the stage to the end of the limit switches, then locating the center (home) position.

COMMAND SYNTAX	DESCRIPTION
<code>:move:load devID</code>	Moves the stage to the load-device position. The load position is either the front center or front left corner, depending on your model of probe station.
<code>:move:prob:abs:die devID x y</code> <code>:move:prob:abs:die? devID</code>	Moves the stage to the x, y die index position, as defined in the active probe-plan file. The query returns the current probe plan x, y, and z indexes.
<code>:move:prob:abs:index devID n</code> <code>:move:prob:abs:index? devID</code>	Moves the stage to the <i>n</i> th die site or subsite. Only sites designated for testing in the active probe plan file are counted. The query returns the die-site or subsite number corresponding to the current stage position.
<code>:move:prob:abs:subs devID n</code> <code>:move:prob:abs:subs? devID</code>	Moves the stage to the specified subsite index position, as defined in the active probe plan file. The query returns the current probe plan subsite indexes.
<code>:move:prob:firs:die devID</code>	Moves the stage to the first die site designated for testing in the active probe plan file.
<code>:move:prob:firs:subs devID</code>	Moves the stage to the first subsite on the current die designated for testing in the active probe plan file.

COMMAND SYNTAX	DESCRIPTION
<code>:move:prob:last:die devID</code>	Moves the stage to the last die site designated for testing in the active probe plan file.
<code>:move:prob:last:subs devID</code>	Moves the stage to the last subsite designated for testing in the active probe plan file.
<code>:move:prob:next:die devID</code>	Moves the stage to the next die site designated for testing in the active probe plan file.
<code>:move:prob:next:site devID</code>	Moves the stage to the next site designated for testing in the active probe plan file. The site can be either a die site or a subsite, whichever comes next.
<code>:move:prob:next:subs devID</code>	Moves the stage to the next subsite on the current die designated for testing in the active probe plan file.
<code>:move:prob:pri:die devID</code>	Moves the stage to the prior die site designated for testing in the active probe plan file.
<code>:move:prob:pri:site devID</code>	Moves the stage to the prior site designated for testing in the active probe plan file. The site will be either a die site or a subsite, whichever came last.
<code>:move:prob:pri:subs devID</code>	Moves the stage to the prior subsite on the current die designated for testing in the active probe plan file.

COMMAND SYNTAX	DESCRIPTION
<pre> :move:rel devID x y z [none][none][up] [down] [none] </pre>	Lowers the chuck, moves the stage to the supplied x- and y-axis positions, then raises the chuck to the supplied z-axis coordinate position. Moves are made relative to the current position. Use none as the z-axis parameter when you don't want any vertical chuck movement.
<pre> :move:rel:chuc devID x y z [none][none][up] [down] [none] </pre>	This command is the same as the :move:rel command
<pre> :move:rel:nchuc devID x y z [none][none] [up] [down] [none] </pre>	Moves the stage, without first lowering the chuck, to the supplied x- and y-axis positions, then raises the chuck to the supplied z-axis coordinate position. Moves are made relative to the current position. Use none as the z-axis parameter when you don't want any vertical chuck movement. Returns before the move is complete.
<pre> :move:scan:aax devID [right] [in] [up] [ufast] [left] [out] [down] [vfast] [none] [fast] [medium] [slow] [vslow] [uslow] </pre>	Starts the stage moving in the specified directions. Movement continues until the station receives a :move:stop command. The stage moves at the speed (ufast to uslow) specified by the velocity parameter.

COMMAND SYNTAX	DESCRIPTION
<pre>:move:scan:xax devID [right] [ufast] [left] [vfast] [fast] [medium] [slow] [vslow] [uslow]</pre>	Starts the stage moving in the specified x-axis direction. Movement continues until the station receives a :move:stop command. The stage moves at the speed (ufast to uslow) specified by the velocity parameter.
<pre>:move:scan:yax devID [in] [ufast] [out] [vfast] [fast] [medium] [slow] [vslow] [uslow]</pre>	Starts the stage moving in the specified y-axis direction. Movement continues until the station receives a :move:stop command. The stage moves at the speed (ufast to uslow) specified by the velocity parameter.
<pre>:move:scan:zax devID [up] [ufast] [down] [vfast] [fast] [medium] [slow] [vslow] [uslow]</pre>	Starts the chuck moving in the specified z-axis direction. Movement continues until the station receives a :move:stop command. The stage moves at the speed (ufast to uslow) specified by the velocity parameter.
<pre>:move:sep devID :move:sep? devID</pre>	Lowers the chuck to break the contact between the probes and DUT. The query return TRUE if the chuck is down and FALSE if the chuck is up.
<pre>:move:stop:aax devID</pre>	Stops movement along all axes.
<pre>:move:stop:xax devID</pre>	Stops movement along the x axis; that is, all left and right stage motion.

COMMAND SYNTAX	DESCRIPTION
:move:stop:yax <i>devID</i>	Stops movement along the y axis; that is, all in and out stage motion.
:move:stop:zax <i>devID</i>	Stops movement along the z axis; that is, all vertical chuck motion.
:move:up <i>devID</i> :move:up? <i>devID</i>	Raises the chuck. The query returns TRUE if the chuck is up and FALSE if it is down.
:move:wait <i>devID</i> :move:wait? <i>devID</i>	Waits until the current stage-movement is complete before reading the next command. This command only applies to movement started with the :move:abs:nch and :move:rel:nch commands. The query returns TRUE or FALSE depending on whether or not the move has completed.
:prob:alig <i>devID</i>	Reads the alignment position for a probe plan, moves the stage to it, then prompts you to align the probes with the alignment die. This command must immediately follow a:probeplan:load command.
:prob:del <i>devID</i> "pathname"	GPIB applications only. Erases a probe-plan file from the probe-station computer's hard disk.
:prob:down <i>devID</i> "pathname"	GPIB applications only. Downloads a probe-plan file from the host to the probe-station computer's hard disk.

COMMAND SYNTAX	DESCRIPTION
<pre>:prob:load devID "pathname" [off] [off] [on] [on]</pre> <pre>:prob:load? devID</pre>	Loads a probe-plan file from mass storage and makes the file active. The on/off parameters respectively control the display of the wafer map and whether the map is always the top window. After loading, always align the DUT to the station axes with the :probeplan:align command. The query returns the current probe-plan filename, if one is loaded.
<pre>:prob:mark:die devID n color</pre>	Controls which color marks the <i>n</i>th die in the wafer map.
<pre>:prob:nsub? devID</pre>	Returns the number of subsites marked for testing in the active probe plan file.
<pre>:prob:ntes? devID</pre>	Returns the number of die marked for testing in the active probe plan file.
<pre>:prob:set:pal devID color red green blue</pre>	Defines the color palette available for use with the :prob:mark:die command. Also controls the background color for the die.
<pre>:prob:set:subs devID index x y</pre>	Defines a subsite location (stage position relative to the die site). Use the :move:probeplan:absolute:subsite command to position the stage at the subsite location.

COMMAND SYNTAX	DESCRIPTION
:prob:show <i>devID</i> [on] [on] [on] [off] [off] [off]	Controls the display of the probe-plan wafer map. The on/off parameters control: - Display of wafer map - Display on top - Show die coordinates
:prob:unl <i>devID</i>	Unloads the active probe plan.
:prob:upl <i>devID</i> "pathname"	Uploads a probe-plan file from the probe-station computer's hard disk to the host.
:set:acc <i>devID</i> [xy] [vhigh] [z] [high] [medium] [low] [vlow]	Sets the rate of acceleration for stage and chuck movements.
:set:aux <i>devID</i> [off] [on]	Turns power to the probe-station auxiliary power outlet on or off.
:set:busy? <i>devID</i>	Returns TRUE if the probe station is currently performing an operation and FALSE if it is idle.
:set:comp <i>devID</i> [off] [on]	Turns compensation mode on or off. Same as the :set:comp:mode command. You must use PCS menus or the :set:comp:fact command to set the compensation factor.
:set:comp:fact <i>devID</i> x y	Sets the compensation factors along the x- and y-axes that will be applied to moves when compensation mode is on.

COMMAND SYNTAX	DESCRIPTION
:set:comp:mode <i>devID</i> [on] [off]	Turns compensation mode on or off. Same as the :set:comp:fact command. You must use PCS menus or the :set:comp:factor command to set the compensation factor.
:set:del <i>msec</i> :set:del?	Specifies a time delay in milliseconds after each operation. During the delay, the probe station will not execute any commands. The query returns the current delay time.
:set:display:errors <i>devID</i> [on] [off]	Turns the display of device error messages on or off. CAUTION: If off , the probe station ignores error conditions. Errors are not reported and do not stop probe-station operation.
:set:display:messages <i>devID</i> [on] [off]	Turns the display of device messages on or off.
:set:edg <i>devID</i> [on] [off]	Turns edge sense mode on or off.
:set:edg:over <i>devID microns</i>	Sets the number of microns the chuck will rise after contact is made with the edge sense probe.
:set:edg:swit <i>devID</i> [nopen] [nclosed]	Identifies the noncontacted position of the edge sense probe.

COMMAND SYNTAX	DESCRIPTION
<pre>:set:host <i>devID</i> [on] [off]</pre> <pre>:set:host? <i>devID</i></pre>	GPIB only. If a GPIB switch box is connected, this command closes (on) or opens (off) the switch connecting the Summit and Host connectors. The query returns ON or OFF depending on whether the host is connected or not.
<pre>:set:ligh <i>devID</i> [off] [on]</pre>	Turns the microscope light on or off.
<pre>:set:man <i>devID</i> [off] [on]</pre> <pre>:set:man? <i>devID</i></pre>	Turns manual mode on and off. In manual mode, you use the knobs on the probe station to move the stage. The query returns ON or OFF depending on whether or not manual mode is on.
<pre>:set:msaf <i>devID</i> [on] [off]</pre> <pre>:set:msaf? <i>devID</i></pre>	Turns the move-safety feature on or off. When on, you are prompted to confirm stage movements. The query returns ON or OFF depending on whether move-safety is on or off.
<pre>:set:prec <i>devID</i> [on] [off]</pre>	Enables or disables precise moves. When on, causes the stage position to be corrected every time the chuck is raised or lowered.
<pre>:set:pres <i>devID</i> x y</pre>	Presets the coordinate system for: move:abs stage movements by assigning the specified user coordinates to the current stage location.

COMMAND SYNTAX	DESCRIPTION
:set:pri:acc:mod <i>devID</i>	Sets the probe-station move priority to especially accurate module level.
:set:pri:acc:waf <i>devID</i>	Sets the probe-station move priority to especially accurate wafer level.
:set:pri:stan:mod <i>devID</i>	Sets the probe-station move priority to standard module level.
:set:pri:stan:waf <i>devID</i>	Sets the probe-station move priority to standard wafer level.
:set:qui <i>devID</i> [on] [off]	Turns quiet mode on or off. In quiet mode, the motors shut down after each movement.
:set:sal <i>devID</i> [on] [off] :set:sal? <i>devID</i>	Turns software alignment mode on or off. When on , the software compensates for the rotational difference between the probe station's axes and the actual placement of the DUT. The query returns ON or OFF depending on whether or not software alignment mode is on.
:set:show:comm [on] [off] :set:show:comm?	Causes the Command Server to echo the instructions it receives to the screen. The query returns ON or OFF to reflect the whether or not commands are being displayed.

COMMAND SYNTAX	DESCRIPTION
<code>:set:show:err [on] [off]</code> <code>:set:show:err?</code>	Displays errors as the Command Server generates them. The query returns ON or OFF to reflect whether or not error messages are being displayed.
<code>:set:show:resp [on] [off]</code> <code>:set:show:resp?</code>	Displays responses as the Command Server generates them. The query returns ON or OFF reflecting whether or not command responses are being displayed.
<code>:set:station?</code>	Returns the probe station's device identification number.
<code>:set:time:comm?</code>	Returns the duration that commands and command responses are being displayed.
<code>:set:time:err</code>	Returns the duration that error messages are being displayed.

COMMAND SYNTAX	DESCRIPTION
<pre>:set:trav devID max zup zdown zwafer</pre> <pre>:set:travel? devID</pre> CAUTION: This command is for service applications. Use carefully.	Sets the limit of travel from the center of the probe station along the x- and y-axes and the maximum and minimum chuck heights. Also, sets the distance the chuck drops when operating in wafer mode. The query returns the distance from the origin to the limits of stage travel, the maximum and minimum chuck heights, and the distance the chuck drops before the stage moves with wafer-level priority
<pre>:set:unit devID [metr] [Engl]</pre>	Specifies the measurement units to be used for all input/output fields.
<pre>:set:vac devID [on] [off]</pre>	Turns the central chuck vacuum on or off. Does not affect vacuum on corners or auxiliary stages.
<pre>:set:vel devID [xy] [vhigh] [z] [high] [medium] [low] [vlow]</pre>	Specifies the velocity for stage movement along each axis. Does not apply to moves made in scan mode.
<pre>:set:vir devID [on] [off]</pre> <pre>:set:virt?</pre>	Turns virtual mode on and off. When on, allows the server software to run without any probe-station hardware. Returns TRUE if virtual mode is on and FALSE if virtual mode is off.

COMMAND SYNTAX	DESCRIPTION
:set:wafer <i>devID</i> [on] [off]	When on , allows the stage to move when the chuck is only partially lowered. When off , the chuck must be fully lowered before the stage can move.
:syst:beep	Causes the computer to emit a sound.
:syst:del <i>msec</i>	Specifies a single delay period, during which the probe station will not accept commands from application programs.
:syst:disp " <i>string</i> "	Displays a one-line message on the probe-station screen and waits for you to press a key on the keyboard before continuing to the next task.
:syst:err? <i>devID</i>	Returns the last error recorded by the probe station.
:syst:iden? <i>devID</i>	Returns the device identification string.
:syst:lock <i>devID</i> :syst:lock? <i>devID</i>	Locks out commands from other applications. The query returns a TRUE if the probe station is locked and a FALSE if the probe station is unlocked.
:syst:path?	Returns the path to the PCS application files.
:syst:unl <i>devID</i>	Unlocks the probe station so that it can accept commands from other applications.

COMMAND SYNTAX	DESCRIPTION
<code>:syst:vers? devID</code>	Returns the probe station's firmware release version number.

^a When parameters are stacked, choose one.

:move:absolute

Syntax

```
:move:absolute  devID  x      y      z  
                  [none] [none] [up]  
                  [down]  
                  [none]
```

The short command form is **:move:abs**

Description

This command moves the stage to the specified user coordinates. The probe station lowers the chuck before moving the stage and raises the chuck to the supplied z-axis position after the stage reaches the new location.

The coordinates are relative to the location currently defined as the user origin. You can use the **:set:preset** command to change the origin of the coordinate system, or you can change it with the PCS menu interface.

The **:move:absolute:chuck** and the **:move:absolute** commands are identical in function. For example, the following two commands perform the identical stage movement:

```
:move:abs 1 45 181 5200  
:move:abs:chuck 1 45 181 5200
```

Parameters

devID

Specifies which device on the network to send the command to. Use **1** for Summit 10000-series probe stations and **2** for Summit 12000-series probe stations.

x

Specifies the x-axis user coordinate to which the stage will move when this command is executed. This number can be any integer value in the range from -153000 to +153000 μm (-6024 to +6024 mils) or the character string **none**. If **none**, there is no change in the x-axis position.

The actual limits for the value of this parameter depend on the origin of the user coordinate system. The x coordinate value is interpreted as mils or microns. Use the **:set:unit** command to switch between mils and microns.

y

Specifies the y-axis user coordinate to which the stage will move when this command is executed. The legal values for this parameter are the same as previously described for the x-axis coordinate.

z Specifies the chuck height at the end of the move.

For probe stations with pneumatic chucks, use one of the following character strings: **up**, **down**, or **none**. If **none**, the chuck height remains unchanged.

For probe stations with motorized chucks, this parameter can be any positive integer in the range from 400 to 5400 microns or the character strings **up**, **down**, and **none**.

Examples

This example lowers the chuck, moves to the specified user coordinates, and raises the chuck again. In this example, coordinates are in microns. The query form of the move command verifies the stage position.

```
:move:abs 1 11980 -34 5000
:move:abs? 1
+011980 -000034 5000
```

To avoid having the chuck automatically drop before and raise after the move, use the **:move:absolute:nchuck** command.

Related Commands

```
:move:absolute?
:move:absolute:chuck
:move:absolute:nchuck
:move:relative
:move:wait
:set:preset
:set:unit
```

:move:absolute?

Syntax

:move:absolute? *devID*

The short command form is **:move:abs?**

Description

This query returns the current stage position and the chuck height in user coordinates. Coordinates are returned as six-digit numbers, with preceding 0s to make up the six digits. For example, if the current position is at *x*, *y*, and *z* coordinates 1, -524, and 5000, the query returns the following:

+000001 -000524 +005000.

The coordinates are relative to the location currently defined as the user origin. You can use the **:set:preset** command to change the origin of the coordinate system, or you can change it with the PCS menu interface.

Parameters

devID

Specifies which device on the network to send the command to. Use **1** for Summit 10000-series probe stations and **2** for Summit 12000-series probe stations.

Examples

This example lowers the chuck, moves to the specified user coordinates, and raises the chuck again. You then use the query form of the move command to verify the stage position.

```
:move:abs 1 +11980 -34 5200  
:move:abs?  
1 +011980 -000034 5200
```

To avoid having the chuck automatically drop before and raise after the move, use the **:move:absolute:nchuck** command.

Related Commands

```
:move:absolute  
:move:absolute:chuck  
:move:absolute:nchuck  
:move:relative  
:move:wait  
:set:preset  
:set:unit
```

:move:absolute:chuck

Syntax

```
:move:absolute:chuck  devID  x      y      z
                        [none] [none] [up]
                                      [down]
                                      [none]
```

The short command form is **:move:abs:chuc**

Description

This command moves the stage to the specified user coordinates. The probe station lowers the chuck before moving the stage and raises the chuck to the supplied z-axis position after the stage reaches the new location.

The coordinates are relative to the location currently defined as the user origin. You can use the **:set:preset** command to change the origin of the coordinate system, or you can change it with the PCS menu interface.

The **:move:absolute:chuck** and the **:move:absolute** commands are identical in function. For example, the following two commands perform the identical stage movement:

```
:move:abs 1 45 181 5000
:move:abs:chuck 1 45 181 5000
```

Parameters

devID Specifies which device on the network to send the command to. Use **1** for Summit 10000-series probe stations and **2** for Summit 12000-series probe stations.

x Specifies the x-axis user coordinate to which the stage will move when this command is executed. This number can be any integer value in the range from -153000 to +153000 μm (-6024 to +6024 mils), or the character string **none**. If **none**, there is no change in the x-axis position.

The actual limits for the value of this parameter depend on the origin of the user coordinate system. The x coordinate value is interpreted as mils or microns. Use the **:set:unit** command to switch between mils and microns.

y Specifies the y-axis user coordinate to which the stage will move when this command is executed. The legal values for this parameter are the same as previously described for the x-axis coordinate.

z

Specifies the chuck height at the end of the move.

For probe stations with pneumatic chucks, use one of the following character strings: **up**, **down**, or **none**. If **none**, the chuck height remains unchanged.

For probe stations with motorized chucks, this parameter can be any positive integer in the range from 400 to 5400 microns or the character strings **up**, **down**, and **none**.

Examples

This example lowers the chuck, moves the stage to the specified user coordinates, and raises the chuck again. In this example, coordinates are in microns. The query form of the move command verifies the stage position.

```
:move:abs 1 11980 -34 5000
:move:abs?
1+011980 -000034 5000
```

Related Commands

```
:move:absolute?
:move:absolute:chuck
:move:absolute:nchuck
:move:relative
:move:wait
:set:preset
:set:unit
```

:move:absolute:nchuck

Syntax

```
:move:absolute:nchuck  devID  x      y  
                        [none] [none]
```

The short command form is **:move:abs:nch**

Description

This command moves the stage to the specified user coordinates without first lowering the chuck. The height of the chuck remains constant throughout the move and is set at the *z* parameter at the end of the move.

The coordinates are relative to the user origin last set in the menu interface. You can use the **:set:preset** command to change the origin of the coordinate system, or you can change it with the PCS menu interface.

When you use this command, the probe station assumes that the command is completed as soon as the stage begins moving to its new position. This means that the probe station can begin executing the next command before the stage stops moving to the new location.

If you do not want to execute the next command until the stage stops moving, use the **:move:wait** command.

Parameters

devID

Specifies which device on the network to send the command to. Use **1** for Summit 10000-series probe stations and **2** for Summit 12000-series probe stations.

x

Specifies the x-axis user coordinate to which the stage will move when this command is executed. This number can be any integer value in the range -153000 to +153000 μm (-6024 to 6024 mils), or the character string **none**. If **none**, there is no change in the x-axis position.

The actual limits for the value of this parameter depend on the origin of the user coordinate system. The x coordinate value is interpreted as mils or microns. Use the **:set:unit** command to switch between mils and microns.

y

Specifies the y-axis user coordinate to which the stage will move when this command is executed. Valid values for this parameter are the same as previously described for *x*.

Examples

CAUTION *If you leave the chuck up during stage movements, you can damage probes. Make sure the platen is up before you make a stage movement with the chuck up.*

The following example leaves the chuck up during the slow-velocity stage movement. This example assumes the platen is up.

```
:set:vel 1 slow slow none
:move:abs:nch 1 11980 -68 none
:move:cont? 1
TRUE
:move:wait 1
:move:abs? 1
11980 -68 5400
```

The **:move:contact** query returns TRUE because the chuck is still raised. The wait command tells the probe station to wait until the stage reaches its new position before returning the stage location.

Related Commands

```
:move:absolute
:move:absolute?
:move:absolute:chuck
:move:contact
:move:relative
:move:wait
:set:preset
:set:unit
```


:move:center

Syntax

:move:center *devID*

The short command form is **:move:cent**

Description

This command moves the stage to the center (machine 0,0) of the probe station. If the chuck was raised before the stage movement began, this command lowers the chuck and moves the stage.

You can use the **:move:contact**, **:move:separate**, and **:set:vacuum** commands with this command to control vacuum and chuck status.

Parameters

<i>devID</i>	Specifies which device on the network to send the command to. Use 1 for Summit 10000-series probe stations and 2 for Summit 12000-series probe stations.
--------------	--

Examples

This example centers the stage, then uses the **:move:contact** command to raise the chuck.

```
:move:cent 1  
:move:cont 1
```

Related Commands

:move:contact

:move:home

:move:load

:move:contact

Syntax

:move:contact *devID*

The short command form is **:move:cont**

Description

This command raises the chuck to contact the probes. This command does not turn on or off vacuum to the chuck top.

In general, you use this command after a stage movement, when the chuck was down before the stage moved. If the chuck is down and the probe station receives a move command, the probe station makes the stage movement, but does not raise the chuck after the move is completed. The **:move:contact** command allows you to raise the chuck again.

Parameters

devID

Specifies which device on the network to send the command to. Use **1** for Summit 10000-series probe stations and **2** for Summit 12000-series probe stations.

Examples

This example shows the contact command used with one of the move-probeplan commands. In this example, the probe station lowers the chuck, moves the stage, then repositions the stage at a subsite. Once the stage is correctly located over the subsite, the probe station raises the chuck.

```
:move:sep 1  
:move:prob:next 1  
:move:subs 1 1  
:move:cont 1
```

The following example shows the contact command used with the query to determine if the chuck was raised completely.

```
:move:cont 1  
:move:cont? 1  
TRUE
```

Related Commands

:move:contact?
:move:separate

:move:contact?

Syntax

:move:contact? devID

The short command form is **:move:cont?**

Description

This query returns the character string TRUE if the chuck is completely up. If the chuck is down, the query returns the string FALSE.

Parameters

devID

Specifies which device on the network to send the command to. Use **1** for Summit 10000-series probe stations and **2** for Summit 12000-series probe stations.

Examples

The following example shows the contact command used with the query to determine if the chuck was raised completely.

```
:move:cont  
:move:cont?  
TRUE
```

Related Commands

:move:contact

:move:separate

:move:down

Syntax

:move:down *devID*

Description

This command lowers the chuck. The stage position does not change.

Parameters

devID

Specifies which device on the network to send the command to. Use **1** for Summit 10000-series probe stations and **2** for Summit 12000-series probe stations.

Examples

This example lowers the chuck. This example assumes that the chuck was up initially.

```
:move:down 1
```

The following example shows the down command used with the query to determine if the chuck was lowered completely.

```
:move:down 1  
:move:down? 1  
TRUE
```


Related Commands

:move:contact

:move:down?

:move:up

:move:down?

Syntax

:move:down? *devID*

Description

This query returns the character string TRUE if the chuck is completely down and FALSE if the chuck is up.

Parameters

devID

Specifies which device on the network to send the command to. Use **1** for Summit 10000-series probe stations and **2** for Summit 12000-series probe stations.

Examples

This example shows that the probe is completely down.

```
:move:down 1
:move:down?
TRUE
```

Related Commands

:move:down

:move:up

:move:up?

:move:home

Syntax

:move:home *devID*

Description

This command uses the hardware to locate the center (machine 0,0) of the station. It moves the stage to the end of the limit switches at the left rear of the station, then moves the stage to the center position. Use this command to initialize the probe-station hardware.

This command lowers the chuck to move the stage, then raises it again when it reaches the home location.

This command is different from **:move:center** in that **:move:home** re-initializes the stage, instead of merely moving to the currently identified center location.

This command does not reset the probe station to the power-up state.

Parameters

devID

Specifies which device on the network to send the command to. Use **1** for Summit 10000-series probe stations and **2** for Summit 12000-series probe stations.

Examples

The following example shows the **:move:home** command:

```
:move:home 1
```

Related Commands

:move:center

:move:load

Syntax

:move:load *devID*

Description

This command lowers the chuck, moves the stage to the load position, and opens the Load Device dialog box. The dialog box asks you to choose whether or not to move the stage back to center, and the probe station does not continue with the next command until you choose **Yes** or **No**.

The location of the load position depends on your probe station model. Summit 10000-series probe stations load in the front-right corner, while Summit 12000-series probe stations load in the front center.

If you do not want the probe station to pause until you make a choice in the Load Device Dialog box, you can use the **:set:display:messages** command to turn off dialog boxes. If messages are turned off, the probe station will execute the **:move:load** command by moving the stage to the load position. Then, it will execute the next command.

If you are going to change wafers, don't forget to include **:set:vacuum** commands to turn the chuck vacuum on and off.

Parameters

devID Specifies which device on the network to send the command to. Use **1** for Summit 10000-series probe stations and **2** for Summit 12000-series probe stations.

Examples

In this example, you use the **:move:load** command to load a wafer on the chuck and to move the chuck back to the center of the station.

```
:move:load 1
```

When the probe station receives the command line, it drops the chuck (if it is not already down), and moves the stage to the front of the station. The probe-station screen displays a dialog box asking if you want the stage to return to center. When you are done loading your device, choose **Yes** to move the stage back to the center position. The probe station then moves the stage and raises the chuck.

Related Commands

:move:center
:move:home

:move:probeplan:absolute:die

Syntax

:move:probeplan:absolute:die *devID* *x* *y*

The short command form is **:move:prob:abs:die**

Description

This command moves the stage to the *x*, *y* die index position, as defined in the currently-loaded probe plan.

There is no check to ensure that the target index position is marked for testing in the probe plan. The move takes place regardless, and it is not considered an error if the position is unmarked. The axis directions and origin relate only to the current probe plan.

Before using the move commands, you should define test sites with the Probe Plan Editor menus.

Parameters

<i>devID</i>	Specifies which device on the network to send the command to. Use 1 for Summit 10000-series probe stations and 2 for Summit 12000-series probe stations.
<i>x</i>	Specifies the <i>x</i> die index, defined in the probe plan.

y Specifies the y die index, defined in the probe plan.

Examples

This example tells the probe station to move to an index location. You then use the query form of the command to verify the stage position.

```
:move:prob:abs:die 1 2 3
:move:prob:abs:die? 1
2 3
```

Related Commands

```
:move:probeplan:absolute:die?
:move:probeplan:first:die
:move:probeplan:last:die
:move:probeplan:next:die
:move:probeplan:prior:die
:probeplan:load
```

:move:probeplan:absolute:die?

Syntax

:move:probeplan:absolute:die? *devID*

The short command form is **:move:prob:abs:die?**

Description

This query returns the current probe-plan x, y die indexes. These are the die indexes, as defined in the active probe-plan file.

Parameters

devID Specifies which device on the network to send the command to. Use **1** for Summit 10000-series probe stations and **2** for Summit 12000-series probe stations.

Examples

This example moves the stage to an index location. You then use the query form of the command to verify the stage position.

```
:move:prob:abs:die 1 2 3
:move:prob:abs:die? 1
2 3
```

Related Commands

:move:probeplan:absolute:die
:move:probeplan:first:die
:move:probeplan:last:die
:move:probeplan:next:die
:move:probeplan:prior:die
:probeplan:load

:move:probeplan:absolute:index

Syntax

:move:probeplan:absolute:index *devID* *n*

The short command form is **:move:prob:abs:ind**

Description

This command moves the stage to the *n**th* die site listed in the active probe plan. This command returns an error if the specified site has not been defined.

Before using the move commands, you should define test sites with the Probe Plan Editor menus.

The probe station lowers the chuck, moves to the specified location, then raises the chuck again.

Parameters

- | | |
|--------------|--|
| <i>devID</i> | Specifies which device on the network to send the command to. Use 1 for Summit 10000-series probe stations and 2 for Summit 12000-series probe stations. |
| <i>n</i> | The die-site index number indicates the position of the site in the probe-plan die-site list. The index is a positive integer up to five digits long. |

Examples

The following commands load a probe-plan file and move the stage to the 8th die site listed in the probe plan. The probe station is then queried for the current probe position.

```
:prob:load 1 65ghz_t1.ppd  
:move:prob:abs:index 1 8  
:move:prob:abs:index? 1  
8
```

After loading the probe-plan file, the probe station prompts you to position the probes on the alignment die of your substrate. When the probes are aligned, you can use the move command to go to the site, as shown above.

Related Commands

```
:move:probeplan:absolute:subsite?  
:move:probeplan:absolute:die  
:move:probeplan:absolute:die?  
:move:probeplan:first:die  
:move:probeplan:first:subsite  
:move:probeplan:last:die  
:move:probeplan:next:site  
:move:probeplan:next:subsite  
:move:probeplan:prior:die  
:move:probeplan:prior:site  
:move:probeplan:prior:subsite  
:move:probeplan:absolute:index?  
:probeplan:set:subsite
```

:move:probeplan:absolute:index?

Syntax

:move:probeplan:absolute:index? *devID*

The short command form is **:move:prob:abs:ind?**

Description

This query returns the current die site number in the active probe-plan die-site list.

Error Condition

If moves were made without using the list, the return value is invalid. The last valid die-site number is always returned, unless no probe plan is loaded. If there is no probe plan, the probe station returns the value -1.

Parameters

devID

Specifies which device on the network to send the command to. Use **1** for Summit 10000-series probe stations and **2** for Summit 12000-series probe stations.

Examples

The following commands load a probe-plan file and move the stage to the 8th die site listed in the probe plan. The probe station is then queried for the current probe position.

```
:prob:load 1 65ghz_t1.ppd  
:prob:alig 1  
:move:prob:abs:index 1 8  
:move:prob:abs:index? 1  
8
```

After loading the probe-plan file, the probe station prompts you to position the probes on the alignment die of your substrate. When the probes are aligned, you can use the move command to go to the site, as shown above.

Related Commands

:move:probeplan:absolute:index

:move:probeplan:absolute:subsite

Syntax

:move:probeplan:absolute:subsite *devID n*

The short command form is **:move:prob:abs:subs**

Description

This command moves the stage to the specified subsite on the current die. The subsite itself is an offset of x- and y-axis coordinates from the current die site listed in the active probe-plan file.

Before using the move commands, you should define the subsites with **probeplan:set:subsite** commands. Or, you can define subsites using the Probe Plan Editor menus.

CAUTION *Do not attempt to define subsites both with the **probeplan:set:subsite** command and with menus. Use one or the other.*

The probe station lowers the chuck, moves to the specified location, then raises the chuck.

Error Condition

This command returns an error if the specified subsite has not been defined.

Parameters

<i>devID</i>	Specifies which device on the network to send the command to. Use 1 for Summit 10000-series probe stations and 2 for Summit 12000-series probe stations.
<i>n</i>	Specifies the subsite position in the active probe-plan subsite list. Valid entries are positive integers up to five digits long.

Examples

The **:move:probeplan:absolute:subsite** command is usually used with the **:move:probeplan:next:die** and other die-movement commands.

This example assumes that the chuck is up before stage movements are made. The following commands load a probe plan, move to a die, move to a subsite on the die, delay while testing instruments perform a task, then move to another subsite.

```
:prob:load 1 65ghz_t1.ppd  
:prob:alig 1
```

After you load the probe-plan file, the probe station asks you to position the probes on the alignment die of your substrate. When the probes are aligned, you can use the move-to-first-die and subsite commands, as shown. You can then use the system-delay command to tell the probe station to pause while you perform a test.

```
:move:prob:firs:die 1  
:move:prob:abs:subs 1 1  
:syst:del 2000
```

After performing your testing tasks, you can move to subsequent subsites and make additional tests, as shown with these commands:

Using **:move**
:next:site
would
simplify this
example.

```
:move:prob:next:die 1
:move:prob:abs:subs 1 1
:syst:del 2000
:move:prob:abs:subs 1 2
:syst:del 2000
:move:prob:next:die 1
:move:prob:abs:subs 1 2
:syst:del 2000
:move:prob:abs:subs 1 1
:syst:del 2000
```

During each delay, you can perform testing tasks. In the previous examples, the specified delay time is 2 seconds (2000 milliseconds). The last two command lines show that you do not have to enter subsite index numbers in sequential order.

Related Commands

```
:move:probeplan:absolute:subsite?
:move:probeplan:absolute:die
:move:probeplan:absolute:die?
:move:probeplan:first:die
:move:probeplan:first:subsite
:move:probeplan:last:die
:move:probeplan:next:site
:move:probeplan:next:subsite
:move:probeplan:prior:die
:move:probeplan:prior:site
:move:probeplan:prior:subsite
:move:probeplan:absolute:index
:move:probeplan:absolute:index?
probeplan:set:subsite
```

:move:probeplan:absolute:subsite?

Syntax

:move:probeplan:absolute:subsite? *devID*

The short command form is **:move:prob:abs:subs?**

Description

This query returns the current subsite index as listed in the active probe-plan file.

Parameters

<i>devID</i>	Specifies which device on the network to send the command to. Use 1 for Summit 10000-series probe stations and 2 for Summit 12000-series probe stations.
--------------	--

Examples

This example uses the query to confirm the subsite index number.

```
:move:prob:firs:die 1
:move:prob:abs:subs 1 1
:syst:del 2000
:move:prob:abs:subs? 1
1
```

Related Commands

:move:probeplan:absolute:subsite

:move:probeplan:first:die

Syntax

:move:probeplan:first:die *devID*

The short command form is **:move:prob:firs:die**

Description

This command moves the stage to the first die site marked for testing in the active probe-plan file.

The probe station lowers the chuck, moves to the first die location, then raises the chuck.

Error Condition

A probe plan must be loaded before the probe station can execute this command. This command also assumes that you have adjusted theta for your DUT. If you load a probe plan, but have not yet adjusted theta, this command moves the stage to the location it assumes to be the first die marked for testing. If theta has not been properly adjusted, the new location can be inaccurate.

Parameters

devID

Specifies which device on the network to send the command to. Use **1** for Summit 10000-series probe stations and **2** for Summit 12000-series probe stations.

Examples

In this example, you enter the commands that load a probe plan into the probe station and verify that it was successfully loaded:

```
:prob:load 1 65ghz_t1.ppd
:prob:load? 1
65GHZ_T1.PPD
:prob:alig 1
:move:prob:firs:die 1
```

The next command lines show that you can enter other commands, such as **:move:abs**, in between **:move:probeplan** commands. In the following example, you use the next-die command to return the stage to the next die index listed in the active probe-plan file.

```
:move:abs 1 -181 4500 up
:move:prob:next:die 1
:move:prob:abs:subs 1 2
:move:prob:last:die 1
```


Related Commands

:move:probeplan:absolute:subsite
:move:probeplan:absolute:subsite?
:move:probeplan:absolute:die
:move:probeplan:absolute:die?
:move:probeplan:first:subsite
:move:probeplan:last:die
:move:probeplan:next:site
:move:probeplan:next:subsite
:move:probeplan:prior:die
:move:probeplan:prior:site
:move:probeplan:prior:subsite
:move:probeplan:absolute:index
:move:probeplan:absolute:index?
:probeplan:set:subsite

:move:probeplan:first:subsite

Syntax

:move:probeplan:first:subsite *devID*

The short command form is **:move:prob:firs:subs**

Description

This command moves the stage to the first subsite designated for testing in the active probe-plan file. The subsite itself is an offset of x-and y-axis coordinates from the current die site.

Before using the move commands, you should define the subsites with **:probeplan:set:subsite** commands. Or, you can define subsites using the Probe Plan Editor menus.

CAUTION *Do not attempt to define subsites both with the **:probeplan:set:subsite** command and with the Probe Plan Editor. Use one or the other.*

The probe station lowers the chuck, moves to the specified location, then raises the chuck.

Error Condition

This command returns an error if the specified subsite has not been defined.

Parameters

devID

Specifies which device on the network to send the command to. Use **1** for Summit 10000-series probe stations and **2** for Summit 12000-series probe stations.

Examples

The **:move:probeplan:first:subsite** command is usually used with the **:move:probeplan:next:die** and other die-movement commands.

This example assumes that the chuck is up before stage movements are made. The following commands load a probe plan, move to the first die, move to the first subsite on the die, delay while testing instruments perform a task, then move to another subsite.

```
:prob:load 1 65ghz_t1.ppd
:prob:alig 1
```

After you load the probe-plan file, the probe station asks you to position the probes on the alignment die of your substrate. When the probes are aligned, you can use the move-to-first-die and subsite commands, as shown. You can then use the system-delay command to tell the probe station to pause while you perform a test.

```
:move:prob:firs:die 1
:move:prob:first:subs 1
:syst:del 1 2000
```

After performing your testing tasks, you can move to the next subsite and make another test, as shown with these commands:

```
:move:prob:abs:subs 1 2  
:syst:del 1 2000
```

During each delay, you can perform testing tasks. In this example, the specified delay time is 2 sec. (2000 milliseconds). You do not have to enter subsite index numbers in sequential order for positioning the stage.

Related Commands

```
:move:probeplan:absolute:subsite  
:move:probeplan:absolute:subsite?  
:move:probeplan:absolute:die  
:move:probeplan:absolute:die?  
:move:probeplan:first:die  
:move:probeplan:last:die  
:move:probeplan:next:site  
:move:probeplan:next:subsite  
:move:probeplan:prior:die  
:move:probeplan:prior:site  
:move:probeplan:prior:subsite  
:move:probeplan:absolute:index  
:move:probeplan:absolute:index?  
:probeplan:set:subsite
```

:move:probeplan:last:die

Syntax

:move:probeplan:last:die *devID*

The short command form is **:move:prob:last:die**

Description

This command moves the stage to the last die site designated for testing in the active probe-plan file.

The probe station lowers the chuck, moves to the last die location, then raises the chuck.

Error Condition

This command returns an error if the stage is already positioned at the last index.

Parameters

<i>devID</i>	Specifies which device on the network to send the command to. Use 1 for Summit 10000-series probe stations and 2 for Summit 12000-series probe stations.
--------------	--

Examples

The following command lines show the stage moving from one die to the next, then to the last die marked for testing. Delays allow tests to be performed.

```
:move:prob:next:die 1  
:syst:del 5000  
:move:prob:next:die 1  
:move:prob:next:die 1  
:move:prob:next:die 1  
:syst:del 5000  
:move:prob:last:die 1
```

Related Commands

```
:move:probeplan:absolute:subsite  
:move:probeplan:absolute:subsite?  
:move:probeplan:absolute:die  
:move:probeplan:absolute:die?  
:move:probeplan:first:die  
:move:probeplan:first:subsite  
:move:probeplan:next:site  
:move:probeplan:next:subsite  
:move:probeplan:prior:die  
:move:probeplan:prior:site  
:move:probeplan:prior:subsite  
:move:probeplan:absolute:index  
:move:probeplan:absolute:index?  
:probeplan:set:subsite
```

:move:probeplan:last:subsite

Syntax

:move:probeplan:last:subsite *devID*

The short command form is **:move:prob:last:subs**

Description

This command moves the stage to the last subsite listed in the active probe-plan file.

The probe station lowers the chuck, moves to the last subsite location, then raises the chuck.

Error Condition

This command returns an error if the stage is already positioned at the last index or if there are no defined subsites.

Parameters

<i>devID</i>	Specifies which device on the network to send the command to. Use 1 for Summit 10000-series probe stations and 2 for Summit 12000-series probe stations.
--------------	--

Examples

The following command lines show the stage moving from one subsite to the next, then to the last subsite marked for testing. Delays allow tests to be performed.

```
:move:prob:next:die 1  
:syst:del 5000  
:move:prob:next:subs 1  
:move:prob:next:subs 1  
:move:prob:next:subs 1  
:syst:del 5000  
:move:prob:last:subs 1
```

Related Commands

```
:move:probeplan:absolute:subsite  
:move:probeplan:absolute:subsite?  
:move:probeplan:absolute:die  
:move:probeplan:absolute:die?  
:move:probeplan:first:die  
:move:probeplan:first:subsite  
:move:probeplan:last:die  
:move:probeplan:next:site  
:move:probeplan:next:subsite  
:move:probeplan:prior:die  
:move:probeplan:prior:site  
:move:probeplan:prior:subsite  
:move:probeplan:absolute:index  
:move:probeplan:absolute:index?  
:probeplan:set:subsite
```

:move:probeplan:next:die

Syntax

:move:probeplan:next:die *devID*

The short command form is **:move:prob:next:die**

Description

This command moves the stage to the next die site listed in the active probe-plan file.

The probe station lowers the chuck, moves to the specified location, then raises the chuck.

Error Condition

This command returns an error if the stage is already positioned at the last die to be tested because there is no "next" die.

Parameters

<i>devID</i>	Specifies which device on the network to send the command to. Use 1 for Summit 10000-series probe stations and 2 for Summit 12000-series probe stations.
--------------	--

Examples

This example positions the stage at the next die site listed in the active probe-plan file:

```
:move:prob:next:die 1
```

Related Commands

```
:move:probeplan:absolute:subsite  
:move:probeplan:absolute:subsite?  
:move:probeplan:absolute:die  
:move:probeplan:absolute:die?  
:move:probeplan:first:die  
:move:probeplan:first:subsite  
:move:probeplan:last:die  
:move:probeplan:last:subsite  
:move:probeplan:next:site  
:move:probeplan:next:subsite  
:move:probeplan:prior:die  
:move:probeplan:prior:site  
:move:probeplan:prior:subsite  
:move:probeplan:absolute:index  
:move:probeplan:absolute:index?  
:probeplan:set:subsite
```

:move:probeplan:next:site

Syntax

:move:probeplan:next:site *devID*

The short command form is **:move:prob:next:site**

Description

This command moves the stage to the next die site or subsite (whichever is next) listed in the active probe-plan file.

the probe station lowers the chuck, moves to the specified location, then raises the chuck.

Error Condition

This command returns an error if the stage is already positioned at the last site to be tested because there is no "next" site.

Parameters

devID

Specifies which device on the network to send the command to. Use **1** for Summit 10000-series probe stations and **2** for Summit 12000-series probe stations.

Examples

This example positions the stage at the next test site listed in the active probe-plan file:

```
:move:prob:next:site 1
```

Related Commands

```
:move:probeplan:absolute:subsite  
:move:probeplan:absolute:subsite?  
:move:probeplan:absolute:die  
:move:probeplan:absolute:die?  
:move:probeplan:first:die  
:move:probeplan:first:subsite  
:move:probeplan:last:die  
:move:probeplan:last:subsite  
:move:probeplan:next:die  
:move:probeplan:next:subsite  
:move:probeplan:prior:die  
:move:probeplan:prior:site  
:move:probeplan:prior:subsite  
:move:probeplan:absolute:index  
:move:probeplan:absolute:index?  
:probeplan:set:subsite
```

:move:probeplan:next:subsite

Syntax

:move:probeplan:next:subsite *devID*

The short command form is **:move:prob:next:subs**

Description

This command moves the stage to the next subsite listed in the active probe-plan file.

The probe station lowers the chuck, moves to the specified location, then raises the chuck.

Error Condition

This command returns an error if there are no defined subsites or if the stage is already positioned at the last subsite to be tested because there is no "next" subsite.

Parameters

devID

Specifies which device on the network to send the command to. Use **1** for Summit 10000-series probe stations and **2** for Summit 12000-series probe stations.

Examples

This example shows how to tell the probe station to position the stage at the next subsite listed in the active probe-plan file:

```
:move:prob:next:subs 1
```

Related Commands

```
:move:probeplan:absolute:subsite  
:move:probeplan:absolute:subsite?  
:move:probeplan:absolute:die  
:move:probeplan:absolute:die?  
:move:probeplan:first:die  
:move:probeplan:first:subsite  
:move:probeplan:last:die  
:move:probeplan:last:subsite  
:move:probeplan:next:die  
:move:probeplan:next:site  
:move:probeplan:prior:die  
:move:probeplan:prior:site  
:move:probeplan:prior:subsite  
:move:probeplan:absolute:index  
:move:probeplan:absolute:index?  
:probeplan:set:subsite
```


:move:probeplan:prior:die

Syntax

:move:probeplan:prior:die *devID*

The short command form is **:move:prob:pri:die**

Description

This command moves the stage to the previous (prior) die site listed in the active probe-plan file.

The probe station lowers the chuck, moves to the specified location, then raises the chuck.

Error Condition

This command returns an error if the probes are already located on the first die because there is no "prior" die to move to.

Parameters

<i>devID</i>	Specifies which device on the network to send the command to. Use 1 for Summit 10000-series probe stations and 2 for Summit 12000-series probe stations.
--------------	--

Examples

The following command line moves the stage to the prior die site listed in the currently active probe-plan file:

```
:move:prob:pri:die 1
```

Related Commands

```
:move:probeplan:absolute:subsite  
:move:probeplan:absolute:subsite?  
:move:probeplan:absolute:die  
:move:probeplan:absolute:die?  
:move:probeplan:first:die  
:move:probeplan:first:subsite  
:move:probeplan:last:die  
:move:probeplan:last:subsite  
:move:probeplan:next:die  
:move:probeplan:next:site  
:move:probeplan:next:subsite  
:move:probeplan:prior:site  
:move:probeplan:prior:subsite  
:move:probeplan:absolute:index  
:move:probeplan:absolute:index?  
:probeplan:set:subsite
```

:move:probeplan:prior:site

Syntax

:move:probeplan:prior:site *devID*

The short command form is **:move:prob:pri:site**

Description

This command moves the stage to the previous (prior) die site, or subsite, listed in the active probe-plan file. If the probes are located at the first subsite in the list, the stage moves to the prior die site. Otherwise, the stage moves to the prior subsite.

The probe station lowers the chuck, moves to the specified location, then raises the chuck.

Error Condition

This command returns an error if the probes are already located at the first subsite on the first die because there is no "prior" site to move to.

Parameters

devID

Specifies which device on the network to send the command to. Use **1** for Summit 10000-series probe stations and **2** for Summit 12000-series probe stations.

Examples

The following command line moves the stage to the prior site listed in the currently active probe-plan file.

```
:move:prob:pri:site 1
```

Related Commands

```
:move:probeplan:absolute:subsite  
:move:probeplan:absolute:subsite?  
:move:probeplan:absolute:die  
:move:probeplan:absolute:die?  
:move:probeplan:first:die  
:move:probeplan:first:subsite  
:move:probeplan:last:die  
:move:probeplan:last:subsite  
:move:probeplan:next:die  
:move:probeplan:next:site  
:move:probeplan:next:subsite  
:move:probeplan:prior:die  
:move:probeplan:prior:subsite  
:move:probeplan:absolute:index  
:move:probeplan:absolute:index?  
:probeplan:set:subsite
```

:move:probeplan:prior:subsite

Syntax

:move:probeplan:prior:subsite *devID*

The short command form is **:move:prob:pri:subs**

Description

This command moves the stage to the previous (prior) subsite listed in the active probe-plan file.

The probe station lowers the chuck, moves to the specified location, then raises the chuck.

Error Condition

This command returns an error if the probes are already located on the first subsite listed in the subsite list because there is no "prior" subsite.

Parameters

devID

Specifies which device on the network to send the command to. Use **1** for Summit 10000-series probe stations and **2** for Summit 12000-series probe stations.

Examples

The following command line moves the stage to the prior subsite listed in the currently active probe-plan file.

```
:move:prob:pri:subs 1
```

Related Commands

```
:move:probeplan:absolute:subsite  
:move:probeplan:absolute:subsite?  
:move:probeplan:absolute:die  
:move:probeplan:absolute:die?  
:move:probeplan:first:die  
:move:probeplan:first:subsite  
:move:probeplan:last:die  
:move:probeplan:last:subsite  
:move:probeplan:next:die  
:move:probeplan:next:site  
:move:probeplan:next:subsite  
:move:probeplan:prior:die  
:move:probeplan:prior:site  
:move:probeplan:absolute:index  
:move:probeplan:absolute:index?  
:probeplan:set:subsite
```

:move:relative

Syntax

:move:relative *devID* *x y z*

The short command form is **:move:rel**

Description

This command moves the stage to a position relative to the current stage location, as specified by the x-, y-, and z-axis parameters. The probe station waits for the stage movement to complete before beginning execution of the next command.

If the chuck is up initially, the probe station lowers the chuck, moves the stage, and raises the chuck again. If the chuck is down initially, the probe station does not raise the chuck after the stage movement.

This command is identical to the **:move:relative:chuck** command.

Parameters

devID

Specifies which device on the network to send the command to. Use **1** for Summit 10000-series probe stations and **2** for Summit 12000-series probe stations.

x The coordinate value (in microns or mils) that the stage will move along the x-axis, relative to the current stage position. The value of this coordinate can be in the range +153000 to -153000 μm (+6024 to -6024 mils), or the character string **none**. If **none** is entered there is no change in the x-axis direction.

The actual range of valid coordinate values depends on the current stage position. If you enter a value outside the probe station limit switches, the probe station returns an error.

y The coordinate value (in microns or mils) that the stage will move along the y-axis, relative to the current stage position. Refer to the description of the x parameter for information about the valid range of values for this coordinate.

z Specifies the chuck height at the end of the move relative to the current height

For probe stations with a pneumatic chuck, use one of the following character strings: **up**, **down**, or **none**. If **none**, the chuck height remains unchanged.

For probe stations with a motorized chuck, this parameter can be any positive integer that keeps the chuck in the range from 400 to 5400 microns or the character strings **up**, **down**, and **none**.

Examples

In this example, the first command returns the current stage position. The second command moves the stage to a new position, and, when the move completes, returns the new position coordinates.

```
:move:abs? 1  
-022659 -008430 000500  
:move:rel 1 604 181 100  
:move:abs? 1  
-022055 -008249 000600
```

Related Commands

```
:move:absolute  
:move:absolute?  
:move:relative:chuck  
:move:relative:nchuck
```

:move:relative:chuck

Syntax

:move:relative:chuck *devID* *x y z*

The short command form is **:move:rel:chuc**

Description

This command moves the stage to a position relative to the current stage location, as specified by the x-, y-, and z-axis parameters. The probe station waits for the stage movement to complete before beginning execution of the next command.

If the chuck is up initially, the probe station lowers the chuck, moves the stage, and raises the chuck appropriately.

This command is identical to the **:move:relative** command.

Parameters

<i>devID</i>	Specifies which device on the network to send the command to. Use 1 for Summit 10000-series probe stations and 2 for Summit 12000-series probe stations.
--------------	--

x The coordinate value (in microns or mils) that the stage will move along the x-axis, relative to the current stage position. The value of this coordinate can be in the range +153000 to -153000 μm (+6024 to -6024 mils), or the character string **none**. If **none** is entered, there is no change in *x*.

The actual range of valid coordinate values depends on the current stage position. If you enter a value that would take the stage outside the probe-station limit switches, the probe station returns an error.

y The coordinate value (in microns or mils) that the stage will move along the y-axis, relative to the current stage position. Refer to the description of the *x* parameter for information about the valid range of values for this coordinate.

z Specifies the chuck height at the end of the move relative to the current height

For probe stations with a pneumatic chuck, use one of the following character strings: **up**, **down**, or **none**. If **none**, the chuck height remains unchanged.

For probe stations with a motorized chuck, this parameter can be any positive integer that keeps the chuck in the range from 400 to 5400 microns or the character strings **up**, **down**, and **none**.

Examples

In this example, the first command returns the current stage position. The second command moves the stage to a new position, and, when the move completes, returns the new position coordinates.

```
:move:abs? 1  
-022659 -008430 000500  
:move:rel 1 604 181 100  
:move:abs? 1  
-022055 -008249 000600
```

Related Commands

:move:absolute
:move:absolute?
:move:relative
:move:relative:nchuck

:move:relative:nchuck

Syntax

```
:move:relative:nchuck devID x y  
                        [none] [none]
```

The short command form is **:move:rel:nch**

Description

This command moves the stage to a position relative to the current stage location, as specified by the x-, y-, and z-axis parameters.

This command does not lower the chuck before the stage is moved, and the probe station does not wait for the stage movement to complete before beginning execution of the next command. The chuck height remains unchanged.

Parameters

<i>devID</i>	Specifies which device on the network to send the command to. Use 1 for Summit 10000-series probe stations and 2 for Summit 12000-series probe stations.
--------------	--

x The coordinate value (in microns or mils) that the stage will move along the x-axis, relative to the current stage position. The value of this coordinate can be in the range +153000 to -153000 μm (+6024 to -6024 mils), or the character string **none**. If **none** is entered, there is no change in *x*.

The actual range of valid coordinate values depends on the current stage position. If you enter a value outside the probe-station limit switches, the probe station returns an error.

y The coordinate value (in microns or mils) that the stage will move along the y-axis, relative to the current stage position. Refer to the description of the *x* parameter for information about the valid range of values for this coordinate.

Examples

In this example, the first command returns the current stage position. The second command moves the stage to a new position, and, when the move completes, returns the new position coordinates.

```
:move:abs? 1
-022659 -008430 000401
:move:rel:nch 1 604 181 50
:move:abs? 1
-022055 -008249 000451
```


Related Commands

:move:absolute

:move:absolute?

:move:relative

:move:relative:chuck

:move:scan:aaxis

Syntax

```
:move:scan:aaxis devID[left] [in] [up] [ufast]  
[right] [out] [down] [vfast]  
[fast]  
[medium]  
[slow]  
[vslow]  
[uslow]
```

The short command form is **:move:scan:aax**

Description

Use this command for making theta adjustments.

This command moves the stage continuously in the specified x-, y-, and z-axis directions until a stop command is received or until the limit of travel is reached. This command moves the stage diagonally, in three dimensions, toward one of the corners of the station.

If you enter a **:move:stop** command, the probe station halts the specified stage movement. If you do not use a **:move:stop** command to stop stage movement, the stage stops scanning when it reaches the end of the limit switches.

To move the stage along only one axis, refer to the **:move:scan:xax**, **:move:scan:yax**, and **:move:scan:zax** commands.

Parameters

devID Specifies which device on the network to send the command to. Use **1** for Summit 10000-series probe stations and **2** for Summit 12000-series probe stations.

[left]
[right] Indicates the direction of stage movement along the x-axis. This parameter can have the value **left** or **right**.

Note: When viewed through a microscope, it appears that the probes are moving in the opposite direction.

[in]
[out] Indicates the direction of stage movement along the y-axis. This parameter can have the value **in** or **out**. If you select **out**, this command moves the stage toward the front of the probe station. If you select **in**, this command moves the stage toward the rear of the station.

Note: When viewed through a microscope, it appears that the probes are moving in the opposite direction.

[up]
[down] Indicates the direction of movement along the z-axis. This parameter can have the value **up** or **down**. Up raises the chuck, while down lowers the chuck.

[ufast]
...
[uslow] Specifies the velocity at which the stage will move along the axes. This parameter can have one of the following values: **ufast**, **vfast**, **fast**, **medium**, **slow**, **vslow**, or **uslow**. The default value for this parameter is ultra fast (**ufast**).

Examples

In this example, you load a contact substrate on the rear left corner of the chuck. You do not yet know the coordinates of the substrate. Although the load-device command centers the stage, you must still scan the stage so the appropriate corner of the chuck is under the microscope. This example assumes you have set the appropriate vacuum control knob to the on position.

```
:move:load 1  
:move:scan:aax 1 right in down medium
```

The stage moves to the rear right corner of the station. When the stage reaches the appropriate position, use the following stop command to stop stage movement in all axes:

```
:move:stop:aax 1
```

You can now adjust probe-to-DUT height and perform other tasks.

Related Commands

```
:move:scan:xaxis  
:move:scan:yaxis  
:move:scan:zaxis  
:move:stop:aaxis  
:move:stop:xaxis  
:move:stop:yaxis  
:move:stop:zaxis
```

:move:scan:xaxis

Syntax

```
:move:scan:xaxis devID [right] [ufast]
                        [left]  [vfast]
                        [fast]
                        [medium]
                        [slow]
                        [vslow]
                        [uslow]
```

The short command form is **:move:scan:xax**

Description

Use this command for making theta adjustments.

This command moves the stage continuously in the specified x-axis direction until a stop command is received or until the limit of travel is reached. This command moves the stage horizontally only.

You do not have to enter a stop command for horizontal stage movement. If you do not use a **:move:stop** command to stop stage movement, the stage stops when it reaches the right or left end of the limit switches.

Parameters

devID Specifies which device on the network to send the command to. Use **1** for Summit 10000-series probe stations and **2** for Summit 12000-series probe stations.

[left] Indicates the direction of stage movement along the x-axis. This parameter can have the value **left** or **right**.

Note: When viewed through a microscope, it appears that the probes are moving in the opposite direction.

[ufast] Specifies the velocity at which the stage will move along the axes. This parameter can have one of the following values: **ufast**, **vfast**, **fast**, **medium**, **slow**, **vslow**, or **uslow**. The default value for this parameter is ultra fast (**ufast**).

Examples

In this example, you want to scan horizontally to the next element on your substrate so you set the velocity to ultra slow:

```
:move:scan:xax 1 left uslow
```

The stage moves slowly to the left. When the stage reaches the appropriate position, use the following stop command to halt x-axis stage movement:

```
:move:stop:xax 1
```

You can also use the **:move:stop:aaxis** command to stop all stage movement.

Related Commands

```
:move:scan:aaxis  
:move:scan:yaxis  
:move:scan:zaxis  
:move:stop:aaxis  
:move:stop:xaxis
```

:move:scan:yaxis

Syntax

```
:move:scan:yaxis devID [in] [ufast]
                        [out] [vfast]
                        [fast]
                        [medium]
                        [slow]
                        [vslow]
                        [uslow]
```

The short command form is **:move:scan:yax**

Description

Use this command for making theta adjustments.

This command moves the stage continuously in the specified y-axis direction until a stop command is received or until the limit of travel is reached. This command moves the stage vertically only--toward the front and back of the station.

You do not have to enter a stop command to stop vertical stage movement. If you do not use a **:move:stop** command to stop stage movement, the stage stops moving when it reaches the front or rear limit switch.

Parameters

devID Specifies which device on the network to send the command to. Use **1** for Summit 10000-series probe stations and **2** for Summit 12000-series probe stations.

[in] Indicates the direction of stage movement along the y-axis. This parameter can have the value **in** or **out**. If you select **out**, this command moves the stage toward the front of the probe station. If you select **in**, this command moves the stage toward the rear of the station. The positive direction of the y-axis is toward the front of the station, relative to the view through the microscope.

Note: When viewed through a microscope, it appears that the probes are moving in the opposite direction.

[ufast] Specifies the velocity at which the stage will move along the axes. This parameter can have one of the following values: **ufast**, **vfast**, **fast**, **medium**, **slow**, **vslow**, or **uslow**. The default value for this parameter is ultra fast (**ufast**).

Examples

In this example, you want to scan to the top of your DUT until you locate the alignment die. The scan speed is set to medium.

```
:move:scan:yax 1 out medium
```

The stage moves to the front of the station. When the stage reaches the appropriate position, use the following stop command to stop stage movement, then enable the joystick for exact stage positioning.

```
:move:stop:yax 1
```

Related Commands

:move:scan:aaxis

:move:scan:xaxis

:move:scan:zaxis

:move:stop:aaxis

:move:stop:xaxis

:move:stop:yaxis

:move:stop:zaxis

:move:scan:zaxis

Syntax

```
:move:scan:zaxis  devID [up]    [ufast]  
                  [down]  [vfast]  
                        [fast]  
                        [medium]  
                        [slow]  
                        [vslow]  
                        [uslow]
```

The short command form is **:move:scan:zax**

Description

This command moves the stage continuously in the specified z-axis direction until a stop command is received or until the limit of travel is reached.. This command moves the stage vertically, up and down.

You do not have to enter a stop command to stop vertical stage movement. If you do not use a **:move:stop** command to stop stage movement, the stage stops moving when it reaches the up or down limit switch.

Parameters

devID Specifies which device on the network to send the command to. Use **1** for Summit 10000-series probe stations and **2** for Summit 12000-series probe stations.

[up]	Indicates the direction of movement along the z-axis.
[down]	This parameter can have the value up or down . Up raises the chuck, while down lowers the chuck.
[ufast]	Specifies the velocity at which the stage will move along the axes. This parameter can have one of the following values: ufast , vfast , fast , medium , slow , vslow , or uslow . The default value for this parameter is ultra fast (ufast).
...	
[uslow]	

Examples

In this example, you want to scan up from the stage. The scan speed is set to medium.

```
:move:scan:zax 1 up medium
```

The chuck moves up in the station. When the chuck reaches the appropriate height, use the following stop command to stop stage movement.

```
:move:stop:zax 1
```

Related Commands

```
:move:scan:aaxis
:move:scan:xaxis
:move:scan:yaxis
:move:stop:aaxis
:move:stop:zaxis
```

:move:separate

Syntax

:move:separate *devID*

The short command form is **:move:sep**

Description

This command lowers the chuck to break the contact between the probes and the DUT.

Parameters

devID Specifies which device on the network to send the command to. Use **1** for Summit 10000-series probe stations and **2** for Summit 12000-series probe stations.

Examples

This example uses the separate command to lower the chuck so that the probes are no longer in contact with the wafer (DUT). Then, the query is used to determine if the chuck was lowered.

```
:move:sep 1
:move:sep? 1
TRUE
```

Related Commands

:move:contact
:move:separate?

:move:separate?

Syntax

:move:separate? *devID*

The short command form is **:move:sep?**

Description

This query returns the character string TRUE if the chuck is completely down. If the chuck is up, the query returns the string FALSE.

Parameters

<i>devID</i>	Specifies which device on the network to send the command to. Use 1 for Summit 10000-series probe stations and 2 for Summit 12000-series probe stations.
--------------	--

Examples

This example uses the separate command to lower the chuck so that the probes are no longer in contact with the wafer. Then, the query is used to determine if the chuck was lowered.

```
:move:sep 1
:move:sep? 1
TRUE
```


Related Commands

:move:contact

:move:separate

:move:stop:aaxis

Syntax

:move:stop:aaxis *devID*

The short command form is **:move:stop:aax**

Description

This command stops stage movement along the x-, y-, and z-axes. Use this command to stop stage movement started with **:move:scan** commands.

Parameters

<i>devID</i>	Specifies which device on the network to send the command to. Use 1 for Summit 10000-series probe stations and 2 for Summit 12000-series probe stations.
--------------	--

Examples

This example shows a move-scan command sent to the probe station before the stop command is issued.

```
:move:scan:aaxis 1 left in up uslow  
:syst:del 3000  
:move:stop:aaxis 1
```

The first command line sets the acceleration rate for the scan to be ultra slow. The probe station then scans for 3 seconds. After 3 seconds, the probe station reads the stop-movement command and halts all stage movement along the x-, y- and z-axes

Related Commands

```
:move:scan:aaxis  
:move:scan:xaxis  
:move:scan:yaxis  
:move:scan:zaxis  
:move:stop:xaxis  
:move:stop:yaxis  
:move:stop:zaxis
```

:move:stop:xaxis

Syntax

:move:stop:xaxis *devID*

The short command form is **:move:stop:xax**

Description

This command stops stage movement along the x-axis. If the stage is moving along all axes, this command stops movement along the x-axis and allows the stage to continue moving along the y- and z-axes.

Use this command to stop stage movements started with **:move:scan** commands.

Parameters

<i>devID</i>	Specifies which device on the network to send the command to. Use 1 for Summit 10000-series probe stations and 2 for Summit 12000-series probe stations.
--------------	--

Examples

This example moves the stage along the x-axis only. The first command moves to the user coordinates 294, 551. The second command tells the probe station to stop stage movement along the x-axis.

```
:move:abs 1 -294 -551 down  
:move:scan:xaxis 1 right fast  
:move:stop:xaxis 1
```

The probe station drops the chuck, moves to the specified position (-294, -551), then begins scanning along the x-axis only. Because the stage was scanning only along the x-axis, the stage stops moving when the **:move:stop:xaxis** command is executed. .

Related Commands

```
:move:scan:aaxis  
:move:scan:xaxis  
:move:scan:yaxis  
:move:scan:zaxis  
:move:stop:aaxis  
:move:stop:yaxis  
:move:stop:zaxis
```

:move:stop:yaxis

Syntax

:move:stop:yaxis *devID*

The short command form is **:move:stop:yax**

Description

This command stops stage movement along the y-axis. If the stage is moving along all axes, this command stops movement along the y-axis and allows the stage to continue moving along the x-and z-axes.

Use this command to stop stage movement started with **:move:scan** commands.

Parameters

<i>devID</i>	Specifies which device on the network to send the command to. Use 1 for Summit 10000-series probe stations and 2 for Summit 12000-series probe stations.
--------------	--

Examples

This example moves the stage along both the x- and y-axis. In this example, the commands allow you to locate an element along a horizontal line. First, you need to find the line, then the element along the line.

```
:move:scan:aaxis 1 right in down slow  
:move:stop:yaxis 1  
:move:stop:xaxis 1
```

In the example, the probe station drops the chuck and begins scanning in the *x* direction and the *y* direction. When the horizontal line is located, the y-axis stop command is executed. The stage continues to move along the x-axis. When the element along the line is located, the x-axis stop command is issued.

Related Commands

```
:move:scan:aaxis  
:move:scan:xaxis  
:move:scan:yaxis  
:move:scan:zaxis  
:move:stop:aaxis  
:move:stop:xaxis  
:move:stop:zaxis
```

:move:stop:zaxis

Syntax

:move:stop:zaxis *devID*

The short command form is **:move:stop:zax**

Description

This command stops stage movement along the z-axis, but allows the stage to continue moving along the x- and y-axes.

Use this command to stop stage movements started with **:move:scan** commands.

Parameters

<i>devID</i>	Specifies which device on the network to send the command to. Use 1 for Summit 10000-series probe stations and 2 for Summit 12000-series probe stations.
--------------	--

Examples

This example shows a move-scan command sent to the probe station before the stop command is issued.

```
:move:scan:aaxis 1 right in down uslow  
:syst:del 3000  
:move:stop:zaxis
```

In the example, the first command line sets the acceleration rate for the scan to be ultra slow. The probe station then scans for 3 seconds. After 3 seconds, the probe station reads the stop-movement command and halts all stage movement along the z-axis.

Related Commands

```
:move:scan:aaxis  
:move:scan:xaxis  
:move:scan:yaxis  
:move:stop:aaxis  
:move:stop:xaxis
```

:move:up

Syntax

:move:up *devID*

Description

This command raises the chuck to its maximum height.

This command is the opposite of the **:move:down** command.

You can use the **:move:contact**, **:move:separate**, **:move:center**, and **:set:vacuum** commands with this command to control vacuum and chuck status.

Parameters

<i>devID</i>	Specifies which device on the network to send the command to. Use 1 for Summit 10000-series probe stations and 2 for Summit 12000-series probe stations.
--------------	--

Examples

This example raises the chuck. This example assumes that the chuck was down initially.

```
:move:up 1
```

The following example shows the query form used to determine if the chuck was raised completely.

```
:move:up 1  
:move:up? 1  
TRUE
```

Related Commands

```
:move:center  
:move:contact  
:move:down  
:move:home  
:move:load  
:move:scan:aaxis  
:move:scan:xaxis  
:move:scan:yaxis  
:move:scan:zaxis  
:move:stop:aaxis  
:move:stop:xaxis  
:move:stop:yaxis  
:move:stop:zaxis  
:move:up?
```

:move:up?

Syntax

:move:up? *devID*

Description

This query returns the character string TRUE if the chuck is completely up. Otherwise it returns FALSE.

Parameters

devID Specifies which device on the network to send the command to. Use **1** for Summit 10000-series probe stations and **2** for Summit 12000-series probe stations.

Examples

This example shows that the chuck is completely up.

```
:move:up 1
:move:up? 1
TRUE
```

Related Commands

:move:down
:move:up

:move:wait

Syntax

:move:wait *devID*

Description

This command prevents the probe station from executing the next command until it receives a "COMPLETE" string. This string is sent when a move initiated by a prior stage-movement command is complete.

This command applies only to movement started with the **:move:absoulte:nchuck** and **:move:relative:nchuck** commands.

Parameters

devID Specifies which device on the network to send the command to. Use **1** for Summit 10000-series probe stations and **2** for Summit 12000-series probe stations.

Examples

The wait command tells the probe station to wait until the stage reaches its new position before returning the stage location.

```
:set:vel 1 xy slow
:move:abs:nch 1 11980 -68 none
:move:cont? 1
TRUE
:move:wait 1
:move:abs? 1
```

Related Commands

:move:absolute:nchuck

:move:relative:nchuck

:move:wait?

:move:wait?

Syntax

:move:wait? *devID*

Description

This command checks to see if the last move is complete. It returns **TRUE** if the move is complete and **FALSE** if the stage is still moving.

Parameters

<i>devID</i>	Specifies which device on the network to send the command to. Use 1 for Summit 10000-series probe stations and 2 for Summit 12000-series probe stations.
--------------	--

Examples

The following shows the **:move:wait?** query being sent to a station that has completed moving.

```
:move:wait? 2  
TRUE
```

Related Commands

:move:absolute:nchuck

:move:relative:nchuck

:move:wait

:probeplan:align

Syntax

:probeplan:align *devID*

The short command form is **:prob:align**

Description

You must issue this command immediately after a **:probeplan:load** command.

This command sets the alignment position for a probe plan. First the probe station reads the alignment position from the probe plan and moves the stage to that position. Then, a screen message prompts you to move the stage to the actual alignment position and press **Enter**. By doing this, you allow the probe station to compensate for differences in positioning between the actual DUT and the wafer map in the probe plan.

CAUTION *If you do not use this command and the wafer has been moved since the probe plan was created, **:move:probeplan** commands may not position the stage exactly where you need it to be.*

Parameters

devID Specifies which device on the network to send the command to. Use **1** for Summit 10000-series probe stations and **2** for Summit 12000-series probe stations.

Examples

This example shows a probe plan being loaded and the alignment position being set:

```
:prob:load 1 "sandy\300m_fet.ppd"  
:prob:alig 1
```

Related Commands

```
:probeplan:load  
:probeplan:load?
```

:probeplan:delete

Syntax

This command applies to the GPIB only.

:probeplan:delete *devID* "*pathname*"

The short command form is **:prob:del**

It does not apply to local applications.

Description

This GPIB command deletes a probe-plan file from storage. If the file is located in the current directory, you can enter the filename without a full path. If the file is located in a directory other than the current directory, you must enter the full path name of the file.

Error Condition

This GPIB command returns an error if the specified file cannot be located or deleted.

Parameters

devID

Specifies which device on the network to send the command to. Use **1** for Summit 10000-series probe stations and **2** for Summit 12000-series probe stations.

pathname Specifies the path and filename of the probeplan file you want to delete. File and path names follow the MS-DOS naming conventions. The filename must have a three-character file extension. In general, probeplan files have a *.ppd* extension.

Examples

The following GPIB command line shows how to delete the probeplan file *sandy\test1.ppd*.

```
:prob:del 1 "c:\cmi\sandy\test1.ppd"
```

Related Commands

:probeplan:download
:probeplan:load
:probeplan:upload

:probeplan:download

Syntax

This command applies to the GPIB only.

:probeplan:download *devID* "*pathname*"

The short command form is **:prob:down**

It does not apply to local applications.

Description

This GPIB command downloads a probe-plan file from the host to the probe-station computer. This command does not load the specified file into the probe-station interface or make the file the active probe-plan file.

The download command must be immediately followed by the contents of the probe-plan file, line by line. The last line sent by the host must be:

END OF FILE

Refer to the **:prob:load** command for information about loading a probe-plan file.

The END OF FILE string terminates the download-file operation. The probe station is then ready to receive a new remote instruction.

The host program is responsible for specifying the manner in which the file is downloaded. For example, the host must:

1. open a host file for reading
2. read a line from the file
3. send the line to the specified location
4. close the host file when done

You can also set up your host program so that you can specify different path names for the host and probe station file names.

Error Condition

This GPIB command returns an error if the specified path or filename is not valid. If you try to download a file that has the same name as a file which already exists in memory, the existing file will be overwritten.

Parameters

<i>devID</i>	Specifies which device on the network to send the command to. Use 1 for Summit 10000-series probe stations and 2 for Summit 12000-series probe stations.
<i>pathname</i>	Specifies the path and filename of the probe-plan file you want to download. File and path names follow the MS-DOS naming conventions. The filename must have a three-character file extension. In general, probe-plan files have a <i>.ppd</i> extension.

Examples

The following GPIB command line shows how to download the probeplan file *sandy\test2.ppd*. The destination is the current directory so you do not have to specify the path.

```
:prob:down 1 "sandy\test2.ppd"
```

As soon as you send the download command, you must send the contents of the probe-plan file, line by line. When the probe station receives the last line, which must be **END OF FILE**, the download procedure terminates.

The next example shows how to download the same file in to the subdirectory *prodtest\pp401*. Because you are downloading to a location other than the current directory, you must enter the complete path for the file to be downloaded.

```
:prob:down  
"c:\cascade\prodtest\pb401\test2.ppd"
```

Related Commands

```
:probeplan:delete  
:probeplan:load  
:probeplan:upload
```

:probeplan:load

Syntax

:probeplan:load *devID* "*pathname*" [off] [off]
[on] [on]

The short command form is **:prob:load**

Description

This command loads a probe-plan file from disk into RAM and makes the file active. Once a probe plan has been loaded, you can access it from the menu system or with commands.

Immediately following a **:probeplan:load** command, you must issue a **:probeplan:align** command.

Error Condition

If you try to load a file that is not a probe plan, this command returns an error. An error is also returned if you try to load a file that does not exist or is not located at the specified path.

Parameters

<i>devID</i>	Specifies which device on the network to send the command to. Use 1 for Summit 10000-series probe stations and 2 for Summit 12000-series probe stations.
--------------	--

<i>pathname</i>	Specifies the full path and filename of the probe-plan file you want to load. File and path names follow the MS-DOS naming conventions. The filename must have a three-character file extension. In general, probe-plan files have a <i>.ppd</i> extension.
[off] [on]	The first off/on parameter controls whether or not the wafer map is displayed automatically when the probe plan is loaded.
[off] [on]	The second off/on parameter controls whether or not the probe plan windows are always displayed on top of other windows, even when not the active window.

Examples

The following command line shows how to load the probeplan file *sandy\new_test.ppd*.

```
:prob:load 1 "c:\cmi\sandy\new_test.ppd" on on
```

Related Commands

:probeplan:align
:move:probeplan:absolute:die
:move:probeplan:absolute:die?
:move:probeplan:absolute:index
:move:probeplan:absolute:index?
:move:probeplan:absolute:subsite
:move:probeplan:absolute:subsite?
:move:probeplan:first:die
:move:probeplan:first:subsite
:move:probeplan:last:die
:move:probeplan:last:subsite
:move:probeplan:next:die
:move:probeplan:next:site
:move:probeplan:next:subsite
:move:probeplan:prior:die
:move:probeplan:prior:site
:move:probeplan:prior:subsite
:probeplan:load?

:probeplan:load?

Syntax

:probeplan:load? *devID*

The short command form is **:prob:load?**

Description

This query returns the current probe-plan filename. If no probe plan is loaded, the following character string is returned:

```
@No probe plan loaded
```

Parameters

devID

Specifies which device on the network to send the command to. Use **1** for Summit 10000-series probe stations and **2** for Summit 12000-series probe stations.

Examples

In the following example, you query for the currently-active probe-plan filename.

```
:probeplan:load? 1  
TEST_01.PPD
```

Related Commands

:probeplan:align
:move:probeplan:absolute:die
:move:probeplan:absolute:die?
:move:probeplan:absolute:subsite
:move:probeplan:absolute:subsite?
:move:probeplan:first:die
:move:probeplan:first:subsite
:move:probeplan:last:die
:move:probeplan:last:subsite
:move:probeplan:next:die
:move:probeplan:next:site
:move:probeplan:next:subsite
:move:probeplan:prior:die
:move:probeplan:prior:site
:move:probeplan:prior:subsite
:move:probeplan:site
:move:probeplan:site?
:probeplan:load

:probeplan:mark:die

Syntax

:probeplan:mark:die *devID n color*

The short form of this command is **:prob:mark:die**

Description

This command specifies the color used to identify die sites when you display a probeplan wafer map. For example, use of this command is to display die that pass and die that fail as different colors in the wafer-map display.

The *palette* is a table of colors for screen displays.

There are 256 possible colors in the palette, numbered 0 through 255. The first 16 colors (0 through 15) are permanently assigned. The rest you can define with the **:probeplan:set:palette** command. The defined colors in the palette are as follows:

<u>Index</u>	<u>Color</u>
0	Background color
1	Blue
2	Green
3	Cyan
4	Red
5	Magenta
6	Brown
7	Dull White
8	Dark Gray
9	Light Blue

10	Light Green
11	Light Cyan
12	Light Red
13	Light Magenta
14	Light Yellow
15	Bright White

Parameters

<i>devID</i>	Specifies which device on the network to send the command to. Use 1 for Summit 10000-series probe stations and 2 for Summit 12000-series probe stations.
<i>n</i>	Indicates which of the tested die should be marked. The die number is the index number for the die in the probe plan.
<i>color</i>	Specifies what color the probeplan wafer map uses to mark tested die. Use the index number corresponding to the color you want in the palette. There are 256 possible colors, numbered 0 to 255. The first 16 are permanently defined, as shown previously.

Examples

The following commands define color 100 to be red, then sets the 4th die marked for testing in the probeplan wafer map to be displayed as red.

```
:prob:set:palette 1 100 255 0 0  
:prob:mark:die 1 4 100
```

Related Commands

`:probeplan:set:palette`

:probeplan:nsubsites?

Syntax

:probeplan:nsubsites? *devID*

The short command form is **:prob:nsub?**

Description

This query returns the number of active subsites defined in the probe-plan subsite list.

Use the **:probeplan:set:subsite** command to define additional subsites.

Parameters

<i>devID</i>	Specifies which device on the network to send the command to. Use 1 for Summit 10000-series probe stations and 2 for Summit 12000-series probe stations.
--------------	--

Examples

The following command line requests the number of active subsites. In this case, there are eight subsites in the subsite list.

```
:prob:nsub? 1  
8
```


Related Commands

:probplan:subsite
:probplan:ntested?

:probeplan:ntested?

Syntax

:probeplan:ntested? *devID*

The short command form is **:prob:ntes?**

Description

This query returns the number of die marked for testing in the currently active probe plan. If you have not yet loaded a probe-plan file, this query returns an undefined value.

Parameters

<i>devID</i>	Specifies which device on the network to send the command to. Use 1 for Summit 10000-series probe stations and 2 for Summit 12000-series probe stations.
--------------	--

Examples

This example shows how to request the number of die marked for testing. In this example, you first load a probe-plan file. You then enter the die-tested query. In this example, the number of die currently marked for testing is 181 die.

```
:prob:load 1 "sandy\vb\band\65ghz_t1.ppd"  
:prob:ntes? 1  
181
```

Related Commands

:move:probeplan:absolute:subsite
:move:probeplan:absolute:subsite?
:move:probeplan:absolute:die
:move:probeplan:absolute:die?
:move:probeplan:first:die
:move:probeplan:first:subsite
:move:probeplan:last:die
:move:probeplan:last:subsite
:move:probeplan:next:die
:move:probeplan:next:site
:move:probeplan:next:subsite
:move:probeplan:prior:die
:move:probeplan:prior:site
:move:probeplan:prior:subsite
:move:probeplan:site
:move:probeplan:site?
:probeplan:nsubsites?
:probeplan:set:subsite

:probeplan:set:palette

Syntax

:probeplan:set:palette *devID n red green blue*

The short form of this command is **:prob:set:pal**

Description

Use this command to define the colors in the palette. The relative intensity of red, green, and blue determines the color. Intensity ranges from 0 to 255, with 0 being off and 255 being the brightest intensity.

NOTE: *Do not assign all three intensities the value 0 because this re-creates the background color, resulting in an invisible display. Also, setting all intensities to 255 results in white.*

The *palette* is a table of colors for screen displays.

There are 256 possible colors in the palette, numbered 0 to 255. The first 16 are permanently defined, as shown below. You can assign custom colors to other palette indexes.

<u>Index</u>	<u>Color</u>
0	Background color
1	Blue
2	Green
3	Cyan
4	Red
5	Magenta
6	Brown
7	Dull White
8	Dark Gray
9	Light Blue

10	Light Green
11	Light Cyan
12	Light Red
13	Light Magenta
14	Light Yellow
15	Bright White

Parameters

<i>devID</i>	Specifies which device on the network to send the command to. Use 1 for Summit 10000-series probe stations and 2 for Summit 12000-series probe stations.
<i>n</i>	Use this parameter to specify the index number (that is, location in the palette) of the color you are defining. You can use any number between 16 and 255 . Numbers 0 through 15 are reserved.
<i>red</i>	Sets the intensity of red in the pixels. The intensity scale runs from 0 to 255 , with 0 being no red and 255 being the brightest red.
<i>green</i>	Sets the intensity of green in the pixels. The intensity scale runs from 0 to 255 , with 0 being no green and 255 being the brightest green.
<i>blue</i>	Sets the intensity of blue in the pixels. The intensity scale runs from 0 to 255 , with 0 being no blue and 255 being the brightest blue.

Examples

The following commands define color 100 to be red, then sets 100 as the color to display marked die in the probeplan wafer map.

```
:prob:set:palette 1 100 255 0 0  
:prob:mark:die 1 30 100
```

Related Commands

:probeplan:mark:die

:probeplan:load

:probeplan:show

Syntax

```
:probeplan:show devID [on] [on] [on]  
                  [off] [off] [off]
```

The short command form is **:prob:show**

Description

This command controls the probe-plan wafer map display. You can specify:

- whether or not the wafer map is displayed
- whether or not the wafer-map window is always displayed on top of other windows
- whether or not to display die site coordinates

Parameters

<i>devID</i>	Specifies which device on the network to send the command to. Use 1 for Summit 10000-series probe stations and 2 for Summit 12000-series probe stations.
[on] [off]	The first on/off parameter determines whether or not the wafer map is always displayed. Use on to display the map.

[on]	The second on/off parameter determines whether or not the wafer map is always displayed on top of other windows. Use on to display on top.
[off]	
[on]	The third on/off parameter determines whether the wafer map displays the coordinates for the die site the cursor is on. Use on to display coordinates.
[off]	

Examples

The following examples causes the wafer map to always be displayed on top of other windows, even when it is not the active window, and to display die site coordinates.

```
:prob:show 2 on on on
```

Related Commands

:probeplan:load

:probeplan:mark:die

:probeplan:set:subsite

Syntax

:probeplan:set:subsite *devID index x y*

Description

This command defines a subsite location (stage position relative to the die site). The subsite is defined as x- and y-axis offsets from the die location.

After defining a subsite, use the

:move:probeplan:absolute:subsite command to position the stage at the subsite location.

The short form of this command is:

:prob:set:subs

Parameters

<i>devID</i>	Specifies which device on the network to send the command to. Use 1 for Summit 10000-series probe stations and 2 for Summit 12000-series probe stations.
<i>index</i>	Identifies the specific subsite. You may specify an indefinite number of subsites. Each subsite must have an index number to differentiate it from the other subsites. For example, if you have three subsites, use 0, 1, and 2 as the three subsite index numbers.

- x* The x offset for the subsite. This coordinate value is an offset from the defined die sites in the active probe plan. The values you can assign to this parameter depend on the size of the die.
- y* The y offset for the subsite. This coordinate value is an offset from the defined die sites in the active probe-plan file. The values you can assign to this parameter depend on the size of the die.

Examples

The following commands load the probe-plan file *plan1.ppd* into the probe station from the directory *prod_test*, define subsite number 1, move the stage to the first die marked for testing, then move the stage to the newly defined subsite.

```
:prob:load 1 prod_"test\plan1.ppd"  
:prob:set:subs 1 1 24 -117  
:move:prob:firs:die 1  
:move:prob:firs:subs 1
```

Related Commands

:move:probeplan:absolute:subsite
:move:probeplan:absolute:subsite?
:move:probeplan:absolute:die
:move:probeplan:absolute:die?
:move:probeplan:first:die
:move:probeplan:first:subsite
:move:probeplan:last:die
:move:probeplan:last:subsite
:move:probeplan:next:die
:move:probeplan:next:site
:move:probeplan:next:subsite
:move:probeplan:prior:die
:move:probeplan:prior:site
:move:probeplan:prior:subsite
:move:probeplan:site
:move:probeplan:site?
:probeplan:nsubsites?

:probeplan:unload

Syntax

:probeplan:unload *devID*

The short command form is **:prob:unl**

Description

This command unloads the active probe plan.

Parameters

<i>devID</i>	Specifies which device on the network to send the command to. Use 1 for Summit 10000-series probe stations and 2 for Summit 12000-series probe stations.
--------------	--

Error Condition

An error results if this command is issued when no probe plan has been loaded.

Examples

The following unloads the active probe plan:

```
:prob:unl 1
```

Related Commands

:probeplan:delete
:probeplan:download
:probeplan:load
:probeplan:upload

:probeplan:upload

Syntax

This command applies to the GPIB only.

:probeplan:upload *devID* "*pathname*"

The short command form is **:prob:upl**

It does not apply to local applications.

Description

This GPIB command uploads a probe-plan file from the probe-station computer to the host. Immediately after issuing the upload command, the host must read the contents of the probe-plan file, line by line, until an error or the END OF FILE string occurs.

The host program is responsible for controlling the location of the information that is uploaded. For example, the host can store all received lines in a file, send the lines to a printer, or display them on the host screen.

In general, the host program performs four steps:

1. opens a file in which information can be written
2. uploads a line of the prove-plan file
3. writes that line to the host file
4. closes the file when the contents of the probe plan are transferred

Error Condition

This command returns an error if the specified file or path is invalid.

Parameters

<i>devID</i>	Specifies which device on the network to send the command to. Use 1 for Summit 10000-series probe stations and 2 for Summit 12000-series probe stations.
<i>pathname</i>	Specifies the source path and filename of the probe-plan file you want to upload. File and path names follow the MS-DOS naming conventions. The filename must have a three-character file extension. In general, probe-plan files have a <i>.ppd</i> extension.

Examples

The following GPIB command line shows how to upload the probeplan file *c:\cmi\sandy\test2t.ppd*.

```
:probeplan:upload 1 "c:\cmi\sandy\test2.ppd"
```

The file continues to upload until the **END OF FILE**, string is reached.

Related Commands

:probeplan:download

:probeplan:delete

:probeplan:load

:set:acceleration

Syntax

```
:set:acceleration devID [xy] [vhhigh]  
                        [z]  [high]  
                        [medium]  
                        [low]  
                        [vlow]
```

The short command form is **:set:acc**

Description

This command sets the rate of acceleration for stage and chuck movements on semi-automatic probe stations.

Parameters

<i>devID</i>	Specifies which device on the network to send the command to. Use 1 for Summit 10000-series probe stations and 2 for Summit 12000-series probe stations.
[<i>xy</i>] [<i>z</i>]	Indicates whether this sets the rate of acceleration for <i>x</i> , <i>y</i> (horizontal stage) movements or for <i>z</i> (vertical chuck) movements.
[<i>vhhigh</i>] ... [<i>vlow</i>]	Indicates the rate of acceleration for movements. This parameter can have one of the following values: vhhigh , high , medium , low , or vlow . The default value for this parameter is very high (vhhigh).

Examples

This example shows how to set the x y acceleration parameter to the highest value and z acceleration parameter to the lowest value.

```
:set:acc: 1 xy vhigh  
:set:acc: 1 z vlow
```

Related Commands

:set:velocity

:set:auxiliary

Syntax

:set:auxiliary *devID* [on]
[off]

The short command form is **:set:aux**

Description

This command turns the power to the probe-station auxiliary outlet on or off.

Parameters

<i>devID</i>	Specifies which device on the network to send the command to. Use 1 for Summit 10000-series probe stations and 2 for Summit 12000-series probe stations.
[on]	Specifies whether auxiliary power on the station is on or off . The default setting is off .
[off]	

Examples

This example shows how to turn auxiliary power on.

```
:set:aux 1 on
```

:set:busy?

Syntax

:set:busy? *devID*

Description

This query checks to see if the probe station is currently performing any operations. It returns **TRUE** if the probe station is busy and **FALSE** if it is idle.

Parameters

<i>devID</i>	Specifies which device on the network to send the command to. Use 1 for Summit 10000-series probe stations and 2 for Summit 12000-series probe stations.
--------------	--

Examples

The following shows this command sent to an idle probe station:

```
:set:busy? 2  
FALSE
```

Related Commands

:move:wait?
:system:lock

:set:compensation

Syntax

:set:compensation *devID* [on]
[off]

The short command form is **:set:comp**

Description

Use the PCS menu to set or select a compensation factor.

This command turns compensation mode on or off. When turned on, the currently defined compensation factor is used to correct for thermal expansion or other stage-position variables. The correction is applied to all **:move:absolute**, **:move:relative**, and **:move:probeplan** commands. It is also applied to stage-position queries.

This command functions identically to the **:set:compensation:mode** command.

Generally, you should use the Prober Control Software (PCS) menus to calculate and store compensation factors. Or, you can numerically set the compensation factors with the **:set:compensation:factor** command.

Parameters

<i>devID</i>	Specifies which device on the network to send the command to. Use 1 for Summit 10000-series probe stations and 2 for Summit 12000-series probe stations.
--------------	--

[on]	Specifies whether or not to use the
[off]	compensation mode. This parameter can have
	the value on or off . The default value is off .

Examples

This example turns on compensation.

```
:set:comp 1 on
```

:set:compensation:factor

Syntax

:set:compensation:factor *devID* *xfact* *yfact*

The short command form is **:set:comp:fact**

Description

Use the **:set:comp** command to turn on compensation.

This command numerically sets the compensation factors for movement along the x- and y-axes. The compensation factor is a value by which stage moves are multiplied in order to compensate for thermal expansion and other stage position variables.

If you do not know the necessary compensation factors, you can use PCS menu operations to measure the difference between the nominal and actual position of a test site and calculate the compensation factors.

Parameters

<i>devID</i>	Specifies which device on the network to send the command to. Use 1 for Summit 10000-series probe stations and 2 for Summit 12000-series probe stations.
<i>xfactor</i>	The value used to calculate the distance the stage should move along the x-axis (that is, right or left movement). The nominal distance is multiplied by this value to get the actual distance.

yfactor

The value used to calculate the distance the stage should move along the y-axis (that is, forward or backward movement). The nominal distance is multiplied by this value to get the actual distance.

Error Condition

An error results if the compensation factor causes stage movement parameters to fall outside the limits of stage travel.

Examples

The following example sets the x- and y-axis compensation factors to increase the move distance by 1 percent along each axis, then turns on compensation mode, moves the stage, and queries for the new location. Note that the returned stage location is the same as the nominal location, not the actual position with compensation applied. The compensation factor is also applied to the query so that the use of compensation is not reflected in the display of coordinates.

```
:set:comp:fact 1 1.01 1.01
:set:comp:mode 1 on
:move:abs 1 +11980 -34
:move:abs?
1 +011980 -000034
```

Related Commands

:set:compensation

:set:compensation:mode

all **:move:absolute** commands

all **:move:relative** commands

all **:move:probeplan** commands

:set:compensation:mode

Syntax

:set:compensation:mode *devID* [on]
[off]

The short command form is **:set:comp:mode**

Description

Use the PCS menu to set or select a compensation factor.

This command turns compensation mode on or off. When turned on, the currently defined compensation factor is used to correct for thermal expansion or other stage-position variables. The correction is applied to all **:move:absolute**, **:move:relative**, and **:move:probeplan** commands. It is also applied to stage-position queries.

This command functions identically to the **:set:compensation** command.

Generally, you should use the Prober Control Software (PCS) menus to calculate and store compensation factors. Or, you can numerically set the compensation factors with the **:set:compensation:factor** command.

Parameters

<i>devID</i>	Specifies which device on the network to send the command to. Use 1 for Summit 10000-series probe stations and 2 for Summit 12000-series probe stations.
--------------	--

[on]	Specifies whether or not to use the
[off]	compensation mode. This parameter can have
	the value on or off . The default value is off .

Examples

This example turns on compensation.

```
:set:comp:mode 1 on
```

Related Commands

:set:compensation

:set:compensation:factor

all **:move:absolute** commands

all **:move:relative** commands

all **:move:probeplan** commands

:set:delay

Syntax

:set:delay *milliseconds*

The short command form is **:set:del**

Description

This command specifies a time in milliseconds for the probe station to delay after each operation. During the delay, the probe station does not execute any commands.

Parameters

<i>milliseconds</i>	Specifies the length of time, in milliseconds, which the probe station will delay before executing the next command. You can specify a delay time in the range 0 to 65535 msec (65.5 seconds). By default, the delay is 0 .
---------------------	--

Examples

The first line specifies a delay time of 2 seconds and the second line specifies a delay time of 2 minutes:

```
:set:del 2000  
:set:del 120000
```

Related Commands

`:move:wait`
`:set:delay?`
`:system:delay`

:set:delay?

Syntax

:set:delay?

The short command form is **:set:del?**

Description

This query returns the current delay time. The delay is the number of milliseconds the probe station waits after each operation before executing the next command.

Use the **:set:delay** command to set the delay.

Examples

This example shows a delay time of 2 seconds:

```
:set:del?  
2000
```

The following example shows a delay time of 2 minutes:

```
:set:del?  
120000
```

Related Commands

:move:wait

:set:delay

:system:delay

:set:display:errors

Syntax

:set:display:errors *devID* [on]
[off]

The short command form is **:set:disp:err**

Description

This command displays error messages on the probe-station screen. The message remains until you press **Enter** (or other pre-defined key) on the probe-station keyboard to confirm the message.

CAUTION *If **off**, the probe station ignores error conditions. Errors are not reported and do not stop probe-station operation.*

Parameters

devID Specifies which device on the network to send the command to. Use **1** for Summit 10000-series probe stations and **2** for Summit 12000-series probe stations.

[on] Specifies whether the probe station displays error messages. By default, the display is **off**.
[off]

Examples

This example shows how to turn the display of error messages on.

```
:set:disp:err 1 on
```

Related Commands

:set:display:messages

:system:display

:set:display:messages

Syntax

```
:set:display:messages devID [on]  
                        [off]
```

The short command form is **:set:disp:mess**

Description

This command turns the display of messages on and off. The message remains until you press **Enter**, or other pre-defined key on the probe-station keyboard to confirm the message. This feature allows you to insert prompts, warnings, and other notes into your test program.

Dialog boxes are messages. So, if you turn off messages, dialog boxes that are normally displayed as part of a command's execution are not displayed. For example, the Load Device dialog box would not display when the **:move:load** command executes.

Parameters

<i>devID</i>	Specifies which device on the network to send the command to. Use 1 for Summit 10000-series probe stations and 2 for Summit 12000-series probe stations.
[on] [off]	Specifies whether the display of messages is on or off . By default, the display is off .

Examples

This example shows how to turn the display of messages on:

```
:set:disp:mess 1 on
```

Related Commands

:set:display:errors

:system:display

:set:edgesense

Syntax

:set:edgesense *devID* [*on*]
[*off*]

The short command form is **:set:edg**

Description

This command turns edge sense mode on or off. When **on**, the chuck up position is the point at which the DUT makes contact with an edge sense probe plus any specified overdrive distance. Using edge sense allows makes it easier to raise the chuck to the optimal probing height. You can only use edge sense mode with probe stations configured with motorized chucks.

To set the parameters required to operate in edge sense mode, use the **:set:edgesense:overdrive** and **:set:edgesense:switch** commands.

Parameters

devID Specifies which device on the network to send the command to. Use **1** for Summit 10000-series probe stations and **2** for Summit 12000-series probe stations.

[*on*] Use **on** to turn operate in edge sense mode and
[*off*] **off** to turn off edge sensing.

Error Condition

This command is ignored if it is sent to a probe station without a motorized chuck (that is, with a pneumatic or manual chuck). This command also requires that an edge sense probe be installed.

Examples

The following example sets the noncontacted position of the edge sense probe, the amount of overdrive, and turns edge sense mode on:

```
:set:edg:swit 1 nop  
:set:edg:over 1 2  
:set:edg 1 on
```

Related Commands

:set:edgesense:overdrive
:set:edgesense:switch
:move:contact

:set:edgesense:overdrive

Syntax

:set:edgesense:overdrive *devID microns*

The short command form is **:set:edg:over**

Description

This command sets the distance the chuck rises after contact is made with the edge sense probe. This command affects chuck movement only when the probe station is operating in edge sense mode.

Parameters

<i>devID</i>	Specifies which device on the network to send the command to. Use 1 for Summit 10000-series probe stations and 2 for Summit 12000-series probe stations.
<i>microns</i>	Specifies the distance the chuck will rise after contacting the edge sense probe.

Examples

The following example sets the noncontacted position of the edge sense probe, the amount of overdrive, and turns edge sense mode on:

```
:set:edg:swit 1 nop  
:set:edg:over 1 2
```

```
:set:edg 1 on
```

Related Commands

:set:edgesense:overdrive

:set:edgesense:switch

:move:contact

:set:edgesense:switch

Syntax

```
:set:edgesense:switch devID [nopen]  
                        [nclosed]
```

The short command form is **:set:edg:swit**

Description

This command identifies the noncontacted position of the switch in the edge sense probe. For example, if the switch opens when the DUT touches the probe tip, then the noncontacted position is closed.

Parameters

devID Specifies which device on the network to send the command to. Use **1** for Summit 10000-series probe stations and **2** for Summit 12000-series probe stations.

[nopen] Use **nopen** if the switch is normally open and
[nclosed] **nclosed** if the switch is normally closed.

Examples

The following example sets the noncontacted position of the edge sense probe, the amount of overdrive, and turns edge sense mode on:

```
:set:edg:swit 1 nop  
:set:edg:over 1 2  
:set:edg 1 on
```

Related Commands

:set:edgesense

:set:edgesense:overdrive

:set:host

Syntax

:set:host *devID* [on]
[off]

Description

Use this command for GPIB applications only.

This command closes the switch that connects the probe station to the host in the optional GPIB switch box.

Parameters

devID Specifies which device on the network to send the command to. Use **1** for Summit 10000-series probe stations and **2** for Summit 12000-series probe stations.

[on] Use **on** to connect the probe station to the host through the GPIB switch box and **off** to disconnect.
[off]

Error Condition

If there is no GPIB switch box installed, this command is ignored.

Examples

The following command closes the connection in the GPIB switch box, then queries for the status.


```
:set:host 1 on  
:set:host?  
ON
```

Related Commands

:set:host?

:set:host?

Syntax

:set:host? *devID*

Description

This query returns **ON** if the host is connected through the optional GPIB switch box and **OFF** if the host is disconnected.

Parameters

<i>devID</i>	Specifies which device on the network to send the command to. Use 1 for Summit 10000-series probe stations and 2 for Summit 12000-series probe stations.
--------------	--

Error Condition

An error results if there is no GPIB switch box installed.

Examples

The following command disconnects the host from the probe station in the GPIB switch box, then queries for the status.

```
:set:host 1 off
:set:host?
OFF
```

Related Commands

:set:host

:set:light

Syntax

:set:light *devID* [on]
[off]

The short command form is **:set:ligh**

Description

This command turns the microscope light on or off.

Parameters

devID Specifies which device on the network to send the command to. Use **1** for Summit 10000-series probe stations and **2** for Summit 12000-series probe stations.

[on] Specifies whether the microscope light for the probe-station optics is **on** or **off**. By default, the light is **off**.
[off]

Examples

This example shows how to turn the microscope light on.

```
:set:ligh 1 on
```

:set:manual

Syntax

:set:manual *devID* [**on**]
[**off**]

The short command form is **:set:man**

Description

This command toggles the Probe Station Control software (PCS) from semi-automatic to manual mode. The station must be in manual mode before using the x- and y-axis knobs on the probe station to move the stage. The status window continues to track and display the current stage location while in manual mode.

When you exit manual mode, PCS lowers the chuck, moves the motors momentarily, and raises the chuck again.

Parameters

<i>devID</i>	Specifies which device on the network to send the command to. Use 1 for Summit 10000-series probe stations and 2 for Summit 12000-series probe stations.
[on]	Specifies whether manual mode is on or off . The default is off .
[off]	

Examples

This example shows how to turn on manual mode.

```
:set:man 1 on
```

Related Commands

:set:manual?

:set:manual?

Syntax

:set:manual? *devID*

The short command form is **:set:man?**

Description

This query returns **ON** if the probe station is operating in manual mode and **OFF** if operating in semi-automatic mode.

Parameters

devID

Specifies which device on the network to send the command to. Use **1** for Summit 10000-series probe stations and **2** for Summit 12000-series probe stations.

Examples

This example shows how to turn on manual mode and query for the status:

```
:set:man 1 on  
:set:man?  
ON
```

Related Commands

:set:manual

:set:msafety

Syntax

```
:set:msafety devID [on]  
                [off]
```

The short command form is **:set:msaf**

Description

This command turns the move-safety feature on or off. When move safety is on, a message box appears on the computer screen prompting you to confirm each stage move if the platen is not fully up or the chuck is not fully down. This helps prevent probe damage by keeping them from scraping across the DUT or from crashing into the chuck.

Parameters

<i>devID</i>	Specifies which device on the network to send the command to. Use 1 for Summit 10000-series probe stations and 2 for Summit 12000-series probe stations.
[on] [off]	Specifies whether the move-safety feature is on or off . By default, move safety is on .

Examples

This example turns off the move-safety feature:

```
:set:msaf 1 off
```

Related Commands

:set:msafety?

:set:msafety?

Syntax

:set:msafety? *devID*

The short command form is **:set:msaf?**

Description

This query returns **ON** if move-safety prompting is turned on and **OFF** if no move prompts are being displayed.

Parameters

devID

Specifies which device on the network to send the command to. Use **1** for Summit 10000-series probe stations and **2** for Summit 12000-series probe stations.

Examples

This example turns off the move-safety feature, then queries for the status:

```
:set:msaf 1 off  
:set:msaf? 1  
OFF
```

Related Commands

:set:msafety

:set:precise

Syntax

```
:set:precise devID [on]
               [off]
```

The short command form is **:set:prec**

Description

This command, when turned on, causes the probe station to correct the stage position every time the chuck height changes. Use this command when stage movements need to be especially accurate.

Parameters

<i>devID</i>	Specifies which device on the network to send the command to. Use 1 for Summit 10000-series probe stations and 2 for Summit 12000-series probe stations.
[on]	Sets precision mode for stage movements.
[off]	When set to on , it causes PCS to correct the stage position every time the chuck is raised or lowered. By default, this command is off .

Examples

This example shows how to toggle for the precise moves feature.

```
:set:prec 1 on
```

:set:preset

Syntax

:set:preset *devID* *x* *y*

The short command form is **:set:pres**

Description

This command presets the coordinate system coordinates for **:move:absolute** stage movements by assigning the specified user coordinates to the current stage location. You can assign any valid user coordinates to the current stage position.

Parameters

devID

Specifies which device on the network to send the command to. Use **1** for Summit 10000-series probe stations and **2** for Summit 12000-series probe stations.

x

Identifies the x coordinate that will be assigned to the current stage location. This coordinate can be in the range from -153000 to 153000 μm (-6024 to 6024 mils). The value of the parameter is interpreted as either mils or microns. Use the **:set:unit** command to switch between mils and microns.

By default, the user coordinate system has the same origin as the probe-station machine coordinate system.

y Identifies the y coordinate that will be assigned to the current stage location. The valid range of values for this command is the same as for the x parameter.

Examples

This example sets the user 0,0 location. This example assumes that you have already positioned the stage at the location to which you want to assign the user 0,0. You can use **:move:abs?** to return the current stage location, as shown here:

```
:move:abs? 1
+001158 -003341 000400
:set:pres 1 0 0
:move:abs? 1
+000000 +000000 000400
```

The following example shows how to assign coordinates x=150, y=281 to the current stage location:

```
:set:pres 1 150 281
:move:abs? 1
+000150 +000281 000400
```

Related Commands

:move:absolute
:move:absolute?
:move:absolute:chuck
:move:absolute:nchuck

:set:priority:accuracy:module

Syntax

:set:priority:accuracy:module *devID*

The short command form is **:set:pri:acc:mod**

Description

This command sets the stage-movement priority to especially accurate module-level. With this priority, the stage does not move until the chuck is fully lowered, and the stage position is corrected each time the chuck is lowered and raised.

Parameters

devID

Specifies which device on the network to send the command to. Use **1** for Summit 10000-series probe stations and **2** for Summit 12000-series probe stations.

Examples

The following turns on accurate module-level moves:

```
:set:pri:acc:mod 1
```

Related Commands

:set:priority:accuracy:wafer
:set:priority:standard:module
:set:priority:standard:wafer

:set:priority:accuracy:wafer

Syntax

:set:priority:accuracy:wafer *devID*

The short command form is **:set:pri:acc:waf**

Description

This command sets the stage-movement priority to especially accurate wafer-level. With this priority, the stage begins moving when the chuck is only partially lowered, and the stage position is corrected each time the chuck is lowered and raised.

Parameters

devID

Specifies which device on the network to send the command to. Use **1** for Summit 10000-series probe stations and **2** for Summit 12000-series probe stations.

Examples

The following turns on accurate wafer-level moves:

```
:set:pri:acc:waf 1
```

Related Commands

:set:priority:accuracy:module

:set:priority:standard:module

:set:priority:standard:wafer

:set:priority:standard:module

Syntax

:set:priority:standard:module *devID*

The short command form is **:set:pri:stan:mod**

Description

This command sets the stage-movement priority to standard-accuracy, module-level. With this priority, the stage does not move until the chuck is fully lowered, and the stage position is not corrected when the chuck is lowered and raised.

Parameters

devID

Specifies which device on the network to send the command to. Use **1** for Summit 10000-series probe stations and **2** for Summit 12000-series probe stations.

Examples

The following turns on standard module-level moves:

```
:set:pri:stan:mod 1
```

Related Commands

:set:priority:accuracy:module

:set:priority:accuracy:wafer

:set:priority:standard:wafer

:set:priority:standard:wafer

Syntax

:set:priority:standard:wafer *devID*

The short command form is **:set:pri:stan:waf**

Description

This command sets the stage-movement priority to standard-accuracy, wafer-level. With this priority, the stage begins moving when the chuck is only partially lowered, and the stage position is not corrected when the chuck is lowered and raised.

Parameters

devID

Specifies which device on the network to send the command to. Use **1** for Summit 10000-series probe stations and **2** for Summit 12000-series probe stations.

Examples

The following turns on standard wafer-level moves:

```
:set:pri:stan:waf 1
```

Related Commands

:set:priority:accuracy:module

:set:priority:accuracy:wafer

:set:priority:standard:module

:set:quiet

Syntax

:set:quiet *devID* [on]
[off]

The short command form is **:set:qui**

Description

This command turns quiet mode on or off. When quiet mode is on, the probe station turns the motors off after the stage has moved to a new location to reduce noise in the system.

Parameters

<i>devID</i>	Specifies which device on the network to send the command to. Use 1 for Summit 10000-series probe stations and 2 for Summit 12000-series probe stations.
[on] [off]	Specifies whether quiet mode is on or off. Use the character strings on and off for this parameter. By default, quiet mode is off .

Examples

This example turns on quiet mode.

```
:set:qui 1 on
```

:set:salign

Syntax

:set:salign *devID* [*on*]
 [*off*]

The short command form is **:set:sal**

Description

This command turns on and off software alignment mode. When software alignment is on, PCS software adds offsets to stage movements to compensate for rotational misalignment between the DUT and the probe-station axes. This is an alternative to aligning the hardware by adjusting Theta.

Before turning on software alignment, you must calculate and set the offsets. To do this, use the PCS menus to align via software. These offsets must be recalculated whenever you change the DUT.

Parameters

devID Specifies which device on the network to send the command to. Use **1** for Summit 10000-series probe stations and **2** for Summit 12000-series probe stations.

[*on*] Use **on** to apply software alignment to stage moves, or **off** to turn off software alignment.
[*off*]

Examples

The following turns on software alignment mode:

```
:set:sal 1 on
```

Related Commands

:set:salign?

:set:salign?

Syntax

:set:salign? *devID*

The short command form is **:set:sal?**

Description

This query returns ON if the probe station is operating in software alignment mode and OFF if it is not.

Parameters

devID Specifies which device on the network to send the command to. Use **1** for Summit 10000-series probe stations and **2** for Summit 12000-series probe stations.

Examples

The following turns on software alignment mode, then queries for the status:

```
:set:sal 1 on
:set:sal?
ON
```

Related Commands

:set:salign

:set:show:command

Syntax

:set:show:command [on]
[off]

The short command form is **:set:show:comm**

Description

This command displays each command on the screen as it is received (not executed) by the Command Server. The time each command is displayed depends on the time set with the **:set:time:command** command or with the DDE Server's Setup Options dialog box.

Parameters

[on]	Turns the show-command feature on or off
[off]	(the default setting). When set to off , the screen does not display commands. When set to on , this command the probe station displays each command received from the application program. By default, the display of commands is turned off .

Examples

The examples for this command show how to use the **:set:show** commands for troubleshooting. The commands are especially useful for isolating a communications problem between the

application program and the Command Server or one of the servers.

This first example shows how to turn on the display of commands. In this example, you want to display commands, but not responses.

```
:set:show:comm on  
:set:show:resp off
```

In this next example, you want to make sure that each command is received and executed successfully. To do this, you display both commands and responses.

```
:set:show:comm on  
:set:show:resp on
```

Related Commands

:set:show:error
:set:show:response

:set:show:command?

Syntax

:set:show:command?

The short command form is **:set:show:comm?**

Description

This query returns ON or OFF depending on whether or not commands are being displayed on the probe-station computer screen.

You can use the **:set:show:command** command to turn on or off the display of commands.

Parameters

None

Examples

The following example shows how to get the display duration for commands

```
:set:show:comm?  
10
```

Related Commands

:set:show:command

:set:show:error

:set:show:response

:set:show:error

Syntax

:set:show:error [on]
[off]

The short command form is **:set:show:err**

Description

This command turns on and off the display of error messages. When on, each error message gets echoed to the screen as it is generated.

The display duration can be set with the **:set:time:error** command or with the DDE Server's Setup Options dialog box.

This command, used with other commands, is especially useful in helping determine if there is a problem with communication between the application program and the probe station.

Parameters

[on]	Turns the display of errors on or off . The
[off]	default setting is off .

Examples

The following command line turns on the display of error messages:

```
:set:show:err on
```

Related Commands

:set:show:command

:set:show:response

:set:show:error?

Syntax

:set:show:error?

The short command form is **:set:show:err?**

Description

This query returns ON or OFF depending on whether or not error messages are being displayed on the probe-station computer screen.

You can use the **:set:show:error** command to turn on or off the display of errors.

Parameters

None

Examples

The following example shows how to get the display duration for commands

```
:set:show:err?  
10
```

Related Commands

:set:show:command

:set:show:error

:set:show:response

:set:show:response

Syntax

:set:show:response **[on]**
 [off]

The short command form is **:set:show:resp**

Description

This command turns on and off the display of responses. When on, each probe-station response gets echoed to the screen as it is received by the Command Server.

The time that command responses are displayed is determined by settings made with the **:set:time:command** command or with the DDE Server's Setup Options dialog box.

This command, used with other commands, is especially useful in helping determine if there is a problem with communication between the application program and the probe station.

Parameters

[on]	Turns the show-response feature on or off .
[off]	The default setting is off .

Examples

The following command line turns on the display of responses:

```
:set:show:resp on
```

Related Commands

```
:set:show:command
```

```
:set:show:error
```

:set:show:response?

Syntax

:set:show:response?

The short command form is **:set:show:resp?**

Description

This query returns ON or OFF, depending on whether or not responses are being displayed on the probe-station computer screen.

You can use the **:set:show:response** command to turn on or off the display of responses.

Parameters

None

Examples

The following example shows how to get the display duration for commands

```
:set:show:err?  
10
```

Related Commands

:set:show:command

:set:show:error

:set:show:response

:set:station?

Syntax

:set:station?

The short command form is **:set:stat?**

Description

This query returns the probe station device identification number. Use this query to determine which probe station in the system you are communicating with.

The device identification number for Summit 10000-series probe stations is 1, while the number for Summit 12000-series probe stations is 2.

Examples

The following query line requests the identification number from a device in the system. The returned character (2) indicates that the device is a Summit 12000-series probe station.

```
:set:stat?  
2
```

:set:time:command

Syntax

:set:time:command *seconds*

The short command form is **:set:time:comm**

Description

This command sets the number of seconds that commands and responses remain on the probe-station computer screen.

While this command sets the display time, it does not determine whether or not commands and responses are displayed. Use the **:set:show:command** and **:set:show:response** commands to turn on or off the display.

Parameters

<i>seconds</i>	Specifies the number of seconds that commands and responses are displayed.
----------------	--

Examples

The following shows how to set the number of seconds that commands and responses will be displayed.

```
:set:time:comm 20
```


Related Commands

:set:show:command
:set:show:command?
:set:time:command?
:set:time:error

:set:time:command?

Syntax

:set:time:command?

The short command form is **:set:time:comm?**

Description

This query returns the number of seconds that commands and responses are set to display on the probe-station computer screen.

Parameters

None.

Error Condition

This command returns the duration set for the display. It does not reflect whether or not messages are actually being displayed. The duration set is returned even when the show command function is turned off.

Examples

The following shows how to get the number of seconds that commands will be displayed.

```
:set:time:comm?  
20
```

Related Commands

`:set:show:command`
`:set:show:command?`
`:set:time:command`
`:set:time:error?`

:set:time:error

Syntax

:set:time:error *seconds*

The short command form is **:set:time:err**

Description

This command sets the number of seconds that error messages remain on the probe-station computer screen.

While this command sets the display time, it does not determine whether or not errors are displayed. Use the **:set:show:error** command to turn on or off the display.

Parameters

<i>seconds</i>	Specifies the number of seconds that error messages are displayed.
----------------	--

Examples

The following shows how to set the number of seconds that errors will be displayed.

```
:set:time:err 60
```

Related Commands

`:set:show:command`
`:set:show:command?`
`:set:time:command`
`:set:time:error?`

:set:time:error?

Syntax

:set:time:error?

The short command form is **:set:time:err?**

Description

This query returns the number of seconds that error messages are set to display on the probe-station computer screen.

Parameters

None.

Error Condition

This command returns the duration set for the display. It does not reflect whether or not errors are actually being displayed. The duration set is returned even when the show error function is turned off.

Examples

The following shows how to get the number of seconds that errors will be displayed.

```
:set:time:err?  
60
```

Related Commands

`:set:show:error`

`:set:show:error?`

`:set:time:command?`

:set:travel

Syntax

:set:travel *devID xydist zup zdown zwafer*

The short command form is **:set:trav**

Description

This command enables you to set the limits of travel for the probe station. These limits are set at the factory, and you should not have to change them.

With this command, you can set the horizontal distance from the center to the limit of travel. Since the chuck is symmetrical, this distance is the same for both the x- and y-axis. You can also set three z-axis positions: chuck fully raised, chuck fully lowered, and the distance the chuck drops when operating in wafer mode.

CAUTION *This command is included primarily for service applications. Extreme care should be taken not to define limits that are incompatible with probe station hardware or test programs.*

Parameters

<i>devID</i>	Specifies which device on the network to send the command to. Use 1 for Summit 10000-series probe stations and 2 for Summit 12000-series probe stations.
<i>xydist</i>	Specifies the distance in microns from the center to the limits of stage travel along the x- and y-axes. Valid values depend on the horizontal range of your probe station. The string none can also be used to denote “no change” from the current setting.
<i>zup</i>	Specifies the chuck height in microns when the chuck is fully raised. Valid values depend on the z-axis range of your probe station. The character string none can also be used to denote “no change” from the current setting. If your probe station does not have a motorized chuck, this parameter must be none .
<i>zdown</i>	Specifies the chuck height in microns when the chuck is fully lowered. Valid values depend on the z-axis range of your probe station. The character string none can also be used to denote “no change” from the current setting. If your probe station does not have a motorized chuck, this parameter must be none .

zwafer

Specifies the distance the chuck drops before the stage begins moving when operating in wafer mode. Valid values depend on the z-axis range of your probe station. The character string **none** can also be used to denote “no change” from the current setting. If your probe station does not have a motorized chuck, this parameter must be **none**.

Examples

The following command line sets the limit of travel along the x- and y-axes, sets a distance for the chuck to drop in wafer mode, and leaves the chuck up and chuck down settings unchanged.

```
:set:trav 2 103600 none none 200
```

Related Commands

:set:travel?

:set:priority:standard:wafer

:set:priority:accurate:wafer

all **:move:absolute** commands

:set:travel?

Syntax

:set:travel? *devID*

The short command form is **:set:trav?**

Description

This query returns the distance from the origin to the limits of stage travel. Because the chuck is symmetrical, the distance is the same for both the x- and y-axes. The maximum, minimum, and wafer-mode chuck heights are also returned for probe stations with motorized chucks.

Parameters

devID

Specifies which device on the network to send the command to. Use **1** for Summit 10000-series probe stations and **2** for Summit 12000-series probe stations.

Examples

The following command line requests the distance in microns from the origin for probe station 1. Since probe station 1 is a Summit 10000-series probe station, only an xy-axis value is returned.

```
:set:trav? 1  
76200
```

:set:unit

Syntax

```
:set:unit devID [metric]  
                [English]
```

Description

This command specifies the units of measurement that will be used for all input and output fields. For example, if you set the units of measurement to metric units (the default setting), x- and y-axis coordinates are in microns, and the wafer diameter is in centimeters. If you set the units of measurement to English units, the x- and y-axis coordinate values are in mils and the wafer diameter is in inches.

Parameters

<i>devID</i>	Specifies which device on the network to send the command to. Use 1 for Summit 10000-series probe stations and 2 for Summit 12000-series probe stations.
[<i>metric</i>] [<i>English</i>]	Specifies that metric or English units of measurement will be used for distances and other values. By default, the units are metric. Parameters can be abbreviated to metr and engl .

Examples

This example sets English as the units of measurement.

```
:set:unit 1 English
```

:set:vacuum

Syntax

```
:set:vacuum  devID  [on]  
                  [off]
```

The short command form is **:set:vac**

Description

This command turns the vacuum on or off in the center of the chuck. This command does not affect the corner (or auxiliary-stage) vacuum. Nor does it affect the size of the DUT hold down-area.

The corner vacuum controls and the size control for the center vacuum area are manual. Any vacuum ports not covered by the DUT should be turned off for maximum efficiency.

Parameters

<i>devID</i>	Specifies which device on the network to send the command to. Use 1 for Summit 10000-series probe stations and 2 for Summit 12000-series probe stations.
[on] [off]	Specifies that vacuum to the chuck top is on or off (the default). When set to on , this command turns on the vacuum to the chuck top. When set to off , this command turns the chuck-top vacuum off.

Examples

This example shows how to turn on vacuum to the chuck top:

```
:set:vac 1 on
```

Related Commands

:move:load

:set:velocity

Syntax

```
:set:velocity  devID [xy]  [ufast]  
                [z]      [vfast]  
                        [fast]  
                        [medium]  
                        [slow]  
                        [vslow]  
                        [uslow]
```

The short command form is **:set:vel**

Description

This command specifies the velocity setting for chuck and stage movements along all axes on semi-automatic probe stations. For example, this command affects the velocity used for the **:move:absolute**, **:move:relative**, **:move:up**, and **:move:down** commands.

This command does not apply to moves made in scan mode.

Parameters

<i>devID</i>	Specifies which device on the network to send the command to. Use 1 for Summit 10000-series probe stations and 2 for Summit 12000-series probe stations.
--------------	--

xy	Specifies whether you are setting the velocity for horizontal stage movements (along the x- and y-axes) or for vertical chuck movements (along the z-axis).
z	
[ufast]	Specifies the velocity. This parameter can have one of the following values: ufast , vfast , fast , medium , slow , vslow , or uslow . The default value for this parameter is ultra fast (ufast).
...	
[uslow]	

Examples

This example moves the stage at slow velocity and raises the chuck at medium velocity.

```
:set:vel 2 xy slow
:set:vel 2 z medium
:move:abs 2 7000 7000 750
```

Related Commands

:move:absolute

:move:absolute:chuck

:move:absolute:nchuck

:move:relative

All :move:probeplan commands

:set:virtual

Syntax

```
:set:virtual devID [on]  
[off]
```

The short command form is **:set:virt**

Description

This command puts the probe station in virtual mode, which allows prober control software (PCS), the DDE Server, and the GPIB Server to run on a computer that does not have probe-station hardware connected.

Settings in the *DEVSETUP.INI* file determine which type of probe station is emulated in virtual mode.

Parameters

<i>devID</i>	Specifies which device on the network to send the command to. Use 1 for Summit 10000-series probe stations and 2 for Summit 12000-series probe stations.
[on] [off]	Toggles virtual mode on and off . The default is off .

Example

This example enables virtual mode, then queries to see if virtual mode is actually on.

```
:set:virt 1 on  
:set:virt? 1  
ON
```

:set:virtual?

Syntax

:set:virtual? *devID*

The short command form is **:set:virt?**

Description

This query returns the character string ON if the probe station is operating in virtual mode and OFF if the probe station is not in virtual mode.

Virtual mode allows prober control software (PCS), the DDE Server, the GPIB Server, and other probe-station control software to run on a computer that does not have probe-station hardware connected.

Parameters

<i>devID</i>	Specifies which device on the network to send the command to. Use 1 for Summit 10000-series probe stations and 2 for Summit 12000-series probe stations.
--------------	--

Examples

The following command tells you whether or not the probe station is operating in virtual mode:

```
:set:virt? 1  
OFF
```

Related commands

:set:virtual

:set:wafer

Syntax

:set:wafer *devID* [on]
[off]

The short command form is **:set:waf**

Description

This command turn on and off wafer mode. In wafer mode, the stage moves when the chuck is only partially lowered. When off, the chuck must be fully lowered before the stage can move.

Parameters

<i>devID</i>	Specifies which device on the network to send the command to. Use 1 for Summit 10000-series probe stations and 2 for Summit 12000-series probe stations.
[on] [off]	Toggles wafer mode on or off . By default, this mode is off .

Examples

This example shows the :set:wafer command used before moving to the next subsite.

```
:prob:load 1 "c:\cmi\sandy\test1.ppd"  
:prob:alig 1  
:move:probe:first:die 1  
:move:prob:first:subs 1  
:set:wafer 1 on  
:move:prob:next:sub 1
```

Related Commands

:move:absolute

:move:absolute:chuck

:move:relative

:system:beep

Syntax

:system:beep

The short command form is **:syst:beep**

Description

This command causes the system to emit a sound.

Parameters

None

Examples

The following causes the system to emit a beep sound:

```
:syst:beep
```

:system:delay

Syntax

:system:delay *msec*

The short command form is **:syst:del**

Description

This command makes the probe station wait the specified length of time before executing the next instruction. During this delay period, the probe station does not accept commands from application programs or remote hosts.

This is a one-time delay. If you want to specify a delay for the probe station to wait after the execution of every instruction, use the **:set:delay** command.

Error Condition

For remote control applications, avoid setting a delay that is longer than your timeout values. Consider using serial polls or SRQ handshaking instead.

Parameters

<i>msec</i>	Specifies the length of time, in milliseconds, which the probe station will delay. The delay parameter can be a value in the range 0 to 65535 (65.5 seconds).
-------------	---

Examples

The delay features are useful with the **:move:scan** commands. In this example, you want to scan at slow speeds for 2 sec, then stop stage movement. :

```
:move:scan:xax 1 right slow  
:syst:del 2000  
:move:stop:xax 1
```

Related Commands

:set:delay

:move:wait

:system:display

Syntax

:system:display "*string*"

The short command form is **:syst:disp**

Description

This command displays a one-line message on the probe-station screen. The message remains until you press **Enter**, or another designated key on the probe-station keyboard, to confirm the message.

Use this command to insert prompts, warnings, and other messages into a test program.

Error Condition

If you have a remote control application using timeout values and the time it takes you to respond to the message exceeds the time limit, you will time out the probe station. Consider using serial polls or SQR handshaking instead.

Parameters

string

The ASCII string representing the message you want to display on the probe-station screen. The character string can be up to 50 characters long and must be enclosed in double quotes, for example, "*text here*" is correct.

The double quotes that delineate the string are not displayed on the probe-station screen as part of the string, and you cannot include quotes in the string. Characters following the second double quote are interpreted as a part of your application program or as a part of another command. If you want to include quotes in your message, use single quotes, for example "'*quote here*'" is correct.

Examples

This example shows a message that tells the user to perform a particular test. In this example, the second **:move:relative** command executes as soon as you confirm the message "Measure S11" by pressing a key on the probe station.

```
:move:rel 1 14 0 400
:syst:disp "Measure S11. Press any key."
:move:rel 1 14 0 400
```

:system:error?

Syntax

:system:error? *devID*

The short command form is **:syst:err?**

Description

This query returns the last error recorded by the probe station. The returned error is an ASCII string. If no error has occurred, this query returns an empty string.

This feature is especially useful when troubleshooting your system. For example, if your application program is not working as expected, you might have created an error situation that your program does not recognize. In this case, you can use the return-error feature to see what error occurred. This can help you trace the error to a particular line in your test program.

Chapter 3 shows the GPIB status-reporting structure.

This feature is also useful with GPIB applications using serial poll or SRQ handshaking. When the probe station is set so that it does not echo a "Complete" message or an error back to the host, you can program your host so that it checks the STB register after each remote operation. If the error bit in the STB register is set, the host can ask for an error-description string to be uploaded.

Parameters

devID

Specifies which device on the network to send the command to. Use **1** for Summit 10000-series probe stations and **2** for Summit 12000-series probe stations.

Examples

This example uses the **:system:error?** command to see the last error message that occurred.

```
:syst:err? 1
@The number entered is greater than the upper
limit of 6024
```

In the previous case, the last error recorded indicates that you entered (or sent) coordinates for a stage movement. However, the coordinates were outside the probe-station limit switches.

Related Commands

:set:show:error

:system:identification?

Syntax

:system:identification? *devID*

The short command form is **:syst:iden?**

Description

This query returns the probe-station identification string.

Parameters

<i>devID</i>	Specifies which device on the network to send the command to. Use 1 for Summit 10000-series probe stations and 2 for Summit 12000-series probe stations.
--------------	--

Examples

This example requests the identification string from the probe station. The string returned by your the probe station will differ depending on your the probe-station serial number and version.

```
:syst:iden? 1  
Cascade 10000
```

Related Commands

:system:version?
:set:station?

:system:lock

Syntax

:system:lock *devID*

The short command form is **:syst:lock**

Description

This command locks the probe station so that it accepts only commands from the current application program.

Parameters

<i>devID</i>	Specifies which device on the network to send the command to. Use 1 for Summit 10000-series probe stations and 2 for Summit 12000-series probe stations.
--------------	--

Examples

The following command line locks out communication with all other applications:

```
:syst:lock 1
```

Related Commands

:system:lock?
:system:unlock

:system:lock?

Syntax

:system:lock? *devID*

The short command form is **:syst:lock?**

Description

This query returns the character string TRUE if the probe station is locked and FALSE if the probe station is unlocked. A locked probe-station accepts commands only from the locking application.

NOTE

Since the probe station is not locked (can still receive commands from) against the locking application, this command returns FALSE if the query comes from the locking application, even when the probe station is actually locked to all other applications.

Parameters

<i>devID</i>	Specifies which device on the network to send the command to. Use 1 for Summit 10000-series probe stations and 2 for Summit 12000-series probe stations.
--------------	--

Examples

The following command tells you whether or not the probe station is locked:

```
:syst:lock? 1  
FALSE
```

Related Commands

:system:lock

:system:unlock

:system:path?

Syntax

:system:path?

The short command form is **:syst:path**

Description

This query opens the `c:\windows\win.ini` file and returns the path assigned to the `ApplicationPath` topic in the `Cascade Microtech` section of the file. This should be the path to the Cascade PCS application files.

Parameters

None.

Examples

The following shows getting the path to the cascade program group.

```
:syst:path?  
C:\CASCADE\
```

Related Commands

:system:version?

:system:unlock

Syntax

:system:unlock *devID*

The short command form is **:syst:unl**

Description

This command unlocks the probe station so that it can receive commands from other application programs.

Parameters

<i>devID</i>	Specifies which device on the network to send the command to. Use 1 for Summit 10000-series probe stations and 2 for Summit 12000-series probe stations.
--------------	--

Examples

The following command line unlocks the probe station, permitting communication with other applications:

```
:syst:unlock 1
```

Related Commands

:system:lock

:system:lock?

:system:version?

Syntax

:system:version? *devID*

The short command form is **:syst:vers?**

Description

This query returns the probe-station firmware release version.

Parameters

<i>devID</i>	Specifies which device on the network to send the command to. Use 1 for Summit 10000-series probe stations and 2 for Summit 12000-series probe stations.
--------------	--

Examples

This example shows a sample response when you send the version command to the probe station. In this example, the probe station version number is 2.00.

```
:syst:vers? 1
2.00
```

Related Commands

:system:identification?

5

COMMAND SERVER COMMUNICATIONS

Introduction

The Command Server application allows you to write automated test programs to control your probe station. The Command server accepts commands from the DDE Server, the GPIB Server, or other client program and sends them to the probe station. It also returns responses to the client.

This section contains the following information:

- overview of the Command Server
- a code listing of a simple user program that communicates with the probe station through the Command Server

About the Command Server

The Cascade Command Server provides you with interprocess communication. It gives you the ability to control the probe-station hardware from other application programs. When you activate the Command Server, it initializes any Cascade hardware attached to the probe station and waits to receive a command from another application.

The Command Server does not include a Windows user interface, although some commands can cause messages to be displayed on the screen. For example, the **:set:show:error** command causes errors to be displayed in a Windows message box.

Chapter 4, *Command Dictionary*, lists the commands that the Command Server recognizes.

Command Server Program Example

The following code listings show how to establish communication with the Command Server and send commands to the probe station. The application program is written in the C programming language, using Borland International's C compiler.

Three files are listed:

- cmdemo.c
- cmdemo.h
- cmdem.rc
- cmdem.def

CMDEMO.C Listing

```

/*****
FILE      : CMDEMO.CPP
AUTHOR    : CASCADE MICROTECH, INC.
PURPOSE   : Station Control Command Server
             demonstration application
IMPLEMENT : Under Windows 3.1, using Borland C++ 3.1
NOTES     :
*/

/***** EDIT HISTORY

1.00.00  27 MAR 95 GDH  Created.
*/

/***** PROGRAM PHILOSOPHY
This application demonstrates use of the Cascade
Microtech Command Server
to communicate between a user application and the
Cascade Microtech
hardware functions.

To provide the lowest common denominator, this
application is written
in flat C using standard Windows APIs.

To use this demo code for your own purposes, modify
the commands in
the "IssueCommands" function to fit your needs, and
recompile.

Associated files:
    CMDEMO.H
    CMDEMO.RC
    CMDEMO.DEF
*/

/***** GLOBAL DECLARATIONS */

#ifdef __WINDOWS_H
#include "windows.h"
#endif

/* Header file for this code and .RC file. */
#ifdef __CMDEMO_H
#include "cmdemo.h"
#endif

```

```

/*****

#define STR_LEN 80

/* Declare a type for communicating with the Command
Server */
typedef BOOL (FAR PASCAL *LPFNCMEEXECUTECOMMAND)(
LPSTR, LPSTR );

HINSTANCE          hCmdLib;          /*
Instance handle to Command Server */
LPFNCMEEXECUTECOMMAND lpfnCMExecuteCommand; /*
Pointer to Command Server function */
HINSTANCE          hMainInst;        /*
Instance handle of application */
HWND               hMainWin;         /* Handle
to main window */
BOOL               initialized;      /* System
has been initialized */

/* Main Windows callback procedure. */
long FAR PASCAL _export WinProc( HWND, UINT, UINT,
LONG );

/*===== InitializeLibrary
Initialize the Cascade Microtech Command Server.
*/

BOOL InitializeLibrary()
{
typedef BOOL (FAR PASCAL *LPFNCMINITIALIZE)( BOOL );
LPFNCMINITIALIZE lpfnCMInitialize;

BOOL err = FALSE;

hCmdLib = LoadLibrary( "CMMAIN.DLL" );

err = hCmdLib <= HINSTANCE_ERROR;

/* Get the address of the Command Server
initialization function. */
if( !err ) {
(LPFNCMINITIALIZE)lpfnCMInitialize =
GetProcAddress( hCmdLib, "CMInitialize" );
err = !lpfnCMInitialize;
}

/* Get the address of the Command Server executer
function. */
if( !err ) {

```

```

/*****
(LPFNCMEEXECUTECOMMAND)lpfnCMExecuteCommand =
GetProcAddress( hCmdLib, "CMExecuteCommand" );
err = !lpfnCMExecuteCommand;
}

/* Call the library initialization function. */
if( !err ) err = !( lpfnCMInitialize( FALSE ) );

if( err ) MessageBox( GetActiveWindow(), "Could not
initialize CMI Command Server", "Command Server
Error", MB_OK | MB_ICONHAND );

return err;
} /* InitializeLibrary */

/*===== IssueCommands
Issue a series of commands to the hardware through
the Command Server.
Note that the length of the string returned from the
CMExecuteCommand
function can be up to 80 characters long.
*/

void IssueCommands()
{
char szCmd[ STR_LEN ] = "";
char szDevice[ STR_LEN ] = "";
BOOL err = FALSE;
char szResponse[ STR_LEN ] = "";
char szStation[ STR_LEN ]; /* Active station
device number */

/* Send a series of commands to the Cascade
Command Server. */

/* Determine the device id of the active station.
*/
err = !(lpfnCMExecuteCommand( ":SET:STATION?",
szDevice ) );

/* Send a command to display a message on the
proper video screen. Note
that the response string can be up to eighty
characters long. */
if ( !err ) {
err = !(lpfnCMExecuteCommand( ":SYSTEM:DISPLAY
\"Cascade Command Server Demo\"",
szResponse ) );
err = err || !strcmpi( szResponse, "Complete" );
}

/* Send a command to move to an absolute

```

```

/*****
position. */
if ( !err ) {
    wsprintf( szCmd, "%s %s %s", ":MOVE:ABSOLUTE",
szDevice, "50000 50000 UP" );
    if ( !err ) err = !( lpfnCMExecuteCommand( szCmd,
szResponse ) );
    err = err || lstrcmpi( szResponse, "Complete" );
}

    /* Send a command to move to an absolute
position. */
if ( !err ) {
    wsprintf( szCmd, "%s %s %s", ":MOVE:ABSOLUTE",
szDevice, "-60000 -60000 DOWN" );
    if ( !err ) err = !( lpfnCMExecuteCommand( szCmd,
szResponse ) );
    err = err || lstrcmpi( szResponse, "Complete" );
}
} /* IssueCommands */

/*----- WinProc
Called from Windows when a message for this
application is available.
*/

long FAR PASCAL _export WinProc( HWND hMainWin,
                                UINT message,
                                UINT wParam,
                                LONG lParam )
{
    BOOL err = FALSE;

    switch( message ) {
        case WM_CREATE:;
            err = InitializeLibrary();
            if( err ) SendMessage( hMainWin, WM_DESTROY, 0,
0L );
            return 0;

        case WM_COMMAND:
            switch ( wParam ) {
                case IDM_STARTDEMO:
                    InvalidateRect( hMainWin, NULL, TRUE );
                    UpdateWindow( hMainWin );

                    IssueCommands();

                    InvalidateRect( hMainWin, NULL, TRUE );
                    UpdateWindow( hMainWin );
                    return 0;

                case IDM_EXITDEMO:

```

```

/*****
    SendMessage( hMainWin, WM_DESTROY, 0, 0L );
    return 0;
} /* Switch wParam */

case WM_DESTROY:
    FreeLibrary( hCmdLib );
    PostQuitMessage( 0 );
    return 0;
} /* Switch message */

return DefWindowProc( hMainWin, message, wParam,
lParam );
} /* WinProc */

/*===== WinMain
Windows entry point.
*/

int PASCAL WinMain( HINSTANCE hInstance,          /*
Handle of this application instance */
                   HINSTANCE hPrevInstance,      /*
Handle of previous instance this application */
                   LPSTR lpCmdLine,              /*
Supplied command line (not used) */
                   int nCmdShow )                /*
Specify how main window is shown */

{
    char szAppName[] = "Command Server Demo";      /*
Application file name */
    char szCaption[] = "PCS Command Server Demo";  /*
Main window caption */
    MSG msg;                                         /* Struction
containing data from Windows app que */

    hMainInst = hInstance;

    /* Register the window and initialize the
application. */

    /* If no previous instance of this application
exists, register a new window class. */
    if ( !hPrevInstance ) {
        WNDCLASS wndClass;                        /* Class
attributes for the main window class */
        wndClass.style
            = CS_HREDRAW | CS_VREDRAW;
        wndClass.lpfnWndProc
            = WinProc;
        wndClass.cbClsExtra
            = 0;
        wndClass.cbWndExtra
            = 0;
        wndClass.hInstance
            = hInstance;
        wndClass.hIcon
            = LoadIcon( hInstance,
MAKEINTRESOURCE( IDI_CMDEMOICON ) );

```

```

/*****
wndClass.hCursor      = LoadCursor( NULL,
IDC_ARROW );
wndClass.hbrBackground = (HBRUSH)GetStockObject(
WHITE_BRUSH );
wndClass.lpszMenuName  = MAKEINTRESOURCE(
IDM_CMDEMOMENU );
wndClass.lpszClassName = szAppName;
RegisterClass( &wndClass );
}

/* Create the main application window. */
hMainWin = CreateWindow( szAppName,          /*
Window class name */
                        szCaption,           /*
Window caption */
                        WS_OVERLAPPEDWINDOW, /*
Window styles */
                        200,                 /*
Initial X position */
                        200,                 /*
Initial Y position */
                        500,                 /*
Initial X size */
                        200,                 /*
Initial Y size */
                        NULL,                /*
Parent window handle */
                        NULL,                /*
Window menu handle */
                        hInstance,          /*
Program instance handle */
                        NULL );             /*
Creation parameters */

/* Set the main window visibility state. */
ShowWindow( hMainWin, SW_SHOWNORMAL );
UpdateWindow( hMainWin );

/* Obtain messages from Windows and process them
in order. */
while ( GetMessage( &msg, NULL, 0, 0 ) ) {
    TranslateMessage( &msg );
    DispatchMessage( &msg );
}

return msg.wParam;
} /* WinMain */

```


CMDEMO.H Listing

```

/*****
FILE      : CMDEMO.H
AUTHOR    : CASCADE MICROTECH, INC.
PURPOSE   : Station Control Command Server client
            application header file
IMPLEMENT : Under Windows 3.1, using Borland C++ 3.1
NOTES     :
*/

/***** EDIT HISTORY

1.00.00  27 MAY 95 GDH  Created.
*/

/***** PROGRAM PHILOSOPHY
Header file for application that demonstrates
implementation of a communication
link between user application and Station Control
Software Command Server.
*/

#ifndef __CMDEMO_H
#define __CMDEMO_H

#define IDI_CMDEMOICON          101

#define IDM_CMDEMOMENU          101
#define IDM_STARTDEMO          102
#define IDM_EXITDEMO           103

#endif

CMDEMO.DEF Listing;-----
-----
; CASCADE Command Server Demo application definition
file
;-----

```

```
/******  
NAME                CMDEMO  
  
DESCRIPTION         'Cascade Command Server demo'  
EXETYPE             WINDOWS  
STUB                'WINSTUB.EXE'  
CODE                PRELOAD MOVEABLE DISCARDABLE  
DATA                PRELOAD MOVEABLE MULTIPLE  
HEAPSIZE            1024  
STACKSIZE           8192
```

CMDEMO.RC Listing

```
/******  
FILE                : CMDEMO.RC  
AUTHOR              : CASCADE MICROTECH, INC.  
PURPOSE             : Station Control Command Server Demo  
resource file  
IMPLEMENT           : Under Windows 3.1, using Borland C++ 3.1  
NOTES               :  
*/  
  
/***** EDIT HISTORY  
  
1.00.00  27 MAR 95 GDH  Created.  
*/  
  
#include <windows.h>  
  
#include "cmdemo.h"  
  
IDM_CMDEMOMENU MENU  
BEGIN  
    MENUITEM "&Start",          IDM_STARTDEMO  
    MENUITEM "E&xit",          IDM_EXITDEMO  
END  
  
IDI_CMDEMOICON ICON  
BEGIN  
    '00 00 01 00 01 00 20 20 10 00 00 00 00 00 E8 02'  
    '00 00 16 00 00 00 28 00 00 00 20 00 00 00 40 00'  
    '00 00 01 00 04 00 00 00 00 00 80 02 00 00 00 00'  
    '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'  
    '00 00 00 00 80 00 00 00 00 00 80 80 00 80 80 00'  
    '00 00 80 00 80 00 80 80 00 00 C0 C0 C0 00 80 80'  
    '80 00 00 00 FF 00 00 FF 00 00 00 FF FF 00 FF 00'  
    '00 00 FF 00 FF 00 FF FF 00 00 FF FF FF 00 00 00'
```

```

/*****
'00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
'00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
'00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
'00 90 90 00 00 00 00 00 00 00 09 09 00 00 00 00'
'00 90 90 00 00 00 00 00 00 00 09 09 00 00 00 00'
'00 90 90 00 00 00 00 00 00 00 09 09 00 00 00 00'
'00 90 90 00 00 00 00 00 00 00 09 09 00 00 00 00'
'09 99 99 00 00 00 00 00 00 00 99 99 90 00 00 00'
'09 99 99 00 00 00 00 00 00 00 99 99 90 00 00 00'
'99 99 99 90 00 00 00 00 00 09 99 99 99 00 00 09'
'09 99 99 09 00 CC CC 00 90 99 99 90 90 00 00 90'
'00 09 00 00 90 CC CC 09 00 00 90 00 09 00 00 00'
'00 09 00 00 00 00 00 00 00 00 00 90 00 00 00 00'
'00 99 90 00 00 00 00 00 00 00 09 99 00 00 00 00'
'00 99 90 00 00 00 00 00 00 00 09 99 00 00 00 00'
'00 99 90 00 00 00 00 00 00 00 09 99 00 00 00 00'
'00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
'00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
'00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
'00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
'00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
'40 00 04 44 40 04 00 00 04 00 04 44 00 00 00 40'
'04 00 04 00 00 04 00 00 04 00 40 00 40 00 00 40'
'00 40 04 00 00 04 00 00 04 00 40 00 40 00 00 40'
'00 40 04 00 00 04 00 00 04 00 40 00 40 00 00 40'
'00 40 04 44 40 04 00 00 04 00 40 00 40 00 00 40'
'00 40 04 00 00 04 00 40 04 00 40 00 40 00 00 40'
'00 40 04 00 00 04 00 40 04 00 40 00 40 00 00 40'
'04 00 04 00 00 04 04 04 04 00 40 00 40 00 00 44'
'40 00 04 44 40 04 40 00 44 00 04 44 00 00 00 00'
'00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
'00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
'00 00 7F FF FF FE 7F FF FF FE 7D 7F FE BE 7D 7F'
'FE BE 7D 7F FE BE 7D 7F FE BE 78 3F FC 1E 78 3F'
'FC 1E 70 1F F8 0E 68 2C 34 16 5E F4 2F 7A 7E FF'
'FF 7E 7C 7F FE 3E 7C 7F FE 3E 7C 7F FE 3E 7F FF'
'FF FE 7F FF FF FE 7F FF FF FE 7F FF FF FE 7F FF'
'FF FE 47 86 FB 8E 5B BE FB 76 5D BE FB 76 5D BE'
'FB 76 5D 86 FB 76 5D BE DB 76 5D BE DB 76 5B BE'
'AB 76 47 86 73 8E 7F FF FF FE 00 00 00 00'

```

END

I N D E X

\$

\$ (dollar sign), preceding
 GPIB meta commands 3:30
\$ preceding GPIB command
 3:3

*

* preceding GPIB command
 3:3

*cls, GPIB command 3:33

*ese, GPIB command 3:33

*ese?, GPIB query 3:33

*esr?, GPIB query 3:33

*idn?, GPIB query 3:33

*opc, GPIB command 3:33

*opc?, GPIB query 3:33

*rst, GPIB command 3:33

*sre, GPIB command 3:33

*sre?, GPIB query 3:33

*stb?, GPIB query 3:33

*stb?, GPIB query example
 3:10

*tst?, GPIB query 3:33

:

: 3:3

: (colon) preceding probe
 station commands 4:2

;

; (semicolon), separating
 multiple commands 3:30

; (semicolon), separating
 probe station commands 4:2

?

? (question mark), indicating
 a query 4:3

@

@ (at sign) character
 preceding error messages
 3:26, 3:29

[

[] (brackets), used to
 indicate a list of choices 4:2

A

abbreviated commands 3:30
abbreviating commands
3:30, 4:3
abort see :move:stop
commands
acceleration
 setting 4:149–4:150
 timeout values affecting
 handshaking 3:20
accuracy of stage moves
 :set:precise 4:185
 :set:wafer 4:236
Activate DDE 2:6
address
 device variable in
 programs 3:13
 incorrect GPIB 3:11
 setting GPIB 3:5, 3:7
align
 DUT see :move:scan
 commands
 probeplan command
 4:118, 4:125
 software 4:197, 4:199
always on top 4:140
application
 string for 2:3
application program
 running from DDE
 Server 2:8
ASCII strings
 responses string-return
 handshaking 3:26, 3:27
at sign
 @ precedes error
 message 2:4, 3:26, 3:29
auxiliary power

turning on and off 4:151

B

beep command 4:238
bits see also status-reporting
structure
 instrument-defined 3:14
block diagram
 broadcast - response 2:4
 command or query -
 response 2:4
 disconnect 2:5
boldface, used to identify a
literal command or
parameter 4:2
brackets [], used to indicate
a list of choices 4:2
bus address
 assigning GPIB 3:5, 3:7
 GPIB 3:6
 incorrect 3:11
busy query 4:152
bytes see also status-
reporting structure
 status-reporting
 structure 3:14
 verifying register-read
 capability 3:10

C

C programming language
 Command Server
 example 5:2
 DDE example 2:12
call to the DDE Server
2:12–2:14
case-sensitive, commands
are not 3:30

- caution
 - damage to chuck during moves 4:30
 - probeplan:\set:\subsite command 4:49, 4:61
 - probeplan:\set:\subsite command 4:53
 - set:\display:\errors off 4:163
- centering the stage 4:31
- change information *iii*
- character string see string
- characters
 - number of meta and probe-station commands abbreviated to 3:30
- chuck see also vertical
- chuck movement
 - height for wafer mode 4:221
 - loading a device
 - vacuum 4:43
 - raising and lowering see also :move:relative and :move:absolute
 - commands
 - :move:absolute:nchuck 4:28
 - :move:down 4:37, 4:39
 - :move:up 4:111, 4:113
 - :set:wafer 4:236
 - using edgesense 4:167, 4:169, 4:171
 - vacuum 4:228
- client
 - defined 2:2
- client server conversation 2:2
- cls (clear status) command 3:33
- CME (command error) bit
 - ESR register 3:14
- cold link
 - interprocess conversion 2:3
- colon, preceding a probe station command 4:2
- colors
 - defining palette 4:137
 - marking die 4:130
- command execution 3:27
- command line, multiple commands on a line 3:30
- command names abbreviated 3:30
- command responses
 - Complete strings 3:26, 3:27
 - errors 3:26, 3:27, 3:29
 - ProbePlan files mistakes in uploading 3:29
 - string-return
 - handshaking 3:26, 3:27
- command sequences
 - abbreviated 3:30
- Command Server
 - communications model 1:6
 - DDE use of 2:11
 - description of 5:2
 - GPIB use of 3:6
 - overview 5:1–5:2

- sample program 5:2
- command sets
 - GPIB protocol
 - commands 3:33
 - meta commands 3:30
- command-error bit
 - ESR register 3:14
- command-response settings recommended 3:27
- commands
 - abbreviated form 3:30, 4:3
 - Activate DDE 2:6
 - boldface, syntax used to indicate a command 4:2
 - case-sensitive, not 3:30
 - characters which precede different instruction sets 3:30
 - complete form 3:30
 - complete response 3:4
 - DDE Server
 - conformance to standards 2:11
 - introduction to 2:11
 - delimiting 4:2
 - direct to probe station in DDE 2:10
 - display query 4:202
 - displaying 4:200
 - duration of display 4:213, 4:215
 - error bit ESR register 3:14
 - GPIB, DOS version 3:8
 - interface clear 3:3
 - long form 3:30
 - many on one command line 4:2
 - meta 3:31–3:32
 - meta settings 3:27
 - meta, syntax 3:30
 - multiple on one command line 3:30, 4:2
 - queries, syntax for 4:3
 - recommended meta-command response settings 3:27
 - separating 3:30
 - sequences of 4:2
 - space, used to delimit commands and parameters 4:2
 - standards compliance 3:2
 - Start! 3:9
 - string-return
 - handshaking 3:26, 3:27
 - summary of 4:3–4:19
 - symbols preceding GPIB 3:3
 - syntax 3:30, 4:2
 - timeout values requiring 3:20
 - turning on responses 3:4
- commandsopec 3:15
- comments
 - in GPIB programs 3:35
- communication
 - client server model 2:2
 - DDE conversations 2:3
 - timeout value 3:8
- communication problems
 - address assignment, incorrect 3:5

- identifying GPIB
 - problems 3:11, 3:12
 - verifying GPIB
 - communication 3:8, 3:11
 - communications
 - overview of 1:6–1:8
 - compensation
 - :set 4:153
 - :set:factor 4:155
 - :set:mode 4:157
 - software alignment 4:197, 4:199
 - complete form of meta and probe-station commands 3:30
 - COMPLETE string
 - OPC or DNE bit generating 3:14
 - Complete string response in string-response handshaking 3:26, 3:27
 - concepts and terminology
 - Command Server 5:2
 - DDE 2:2
 - Config menu
 - Clear response upon new input 3:8
 - Expect DOS 10K syntax 3:8
 - Pause on error 3:7
 - Terminate on error 3:7
 - contact with DUT
 - :move:contact 4:33
 - :move:separate 4:99, 4:101
 - controller 3:3
 - controlling GPIB 3:3
 - conventions
 - command syntax 4:2
 - status-reporting structure 3:13
 - conversation
 - client server 2:2
 - coordinate system
 - assigning user 4:186
 - defining subsites 4:142
 - in absolute moves 4:19–4:30
 - in relative moves 4:80–4:88
 - specifying units of measure 4:226
 - coordinates
 - wafer map display 4:140
 - creating subsites 4:142
 - CSGPIB.INI file 3:35
 - customer service 1:3
- D**
- data
 - identification of 2:3
 - data display
 - DDE options 2:7
 - DDE
 - application examples 2:12
 - code listing 2:12–2:14
 - command summary table 4:3–4:19
 - concepts and terminology 2:1–2:5
 - end session 2:11
 - main steps for using 2:6

- operation in virtual mode 2:11
- overview 2:1–2:5
- server 4:200, 4:202, 4:204, 4:206, 4:210
- starting 2:6
- syntax standards 2:11
- window 4:200, 4:202, 4:204, 4:206, 4:210
- DDE Command Summary 4:3–4:19
- DDE Server
 - communications model 1:6–1:8
 - compatible with management library 1:8
- DdeClientTransaction()
 - return parameter 2:4
- DDEML
 - DDE Server compatible with 1:8
 - use of 2:1
- defining subsite locations 4:142
- delay
 - :move:wait command 4:114
 - :move:wait? command 4:116
 - :set 4:159, 4:239
 - :system 4:239
 - setting delay for every command 4:159
 - setting one-time delay 4:239
 - using serial polls 4:239
 - using SRQ handshaking 4:239
- delete
 - GPIO probeplan command 4:120
- device variable 3:13
- devices
 - loading. See load-device feature 3:11
- device-under-test
 - die sites
 - moving to 4:45, 4:54, 4:58, 4:62, 4:64, 4:68, 4:74
 - loading 4:43, 4:118, 4:125
 - probe contact with
 - :move:contact 4:33
 - :move:separate 4:101
 - :set:precise 4:185
 - :set:wafer 4:236
 - raising and lowering the chuck 4:37, 4:39, 4:111, 4:113
 - setting vacuum 4:228
 - subsites, defining 4:142
 - subsites, moving to 4:53, 4:54, 4:56, 4:61, 4:62, 4:66, 4:72
- dialog box
 - DDE Server 2:6
 - Enter Command for direct DDE control 2:10
 - Execute Commands 2:8
 - GPIO setup 3:7
 - Open Command Files 2:9
 - Setup Options 2:8

die see also :move:probeplan
commands

alignment die in
probeplan 4:118–4:119
moving to sites 4:45,
4:54, 4:58, 4:62, 4:64,
4:68, 4:74
retrieving number
marked for testing 4:133

dimensions

setting units of measure
4:226

direct control of probe
station 2:10

display see also

:set:show:error and

:set:show:command

:errors 4:163, 4:243

:messages 4:165

:system 4:241

:system identification
4:245

using serial polls 4:241

using SRQ handshaking
4:241

display data

DDE options 2:7

displays

:system:display

command 3:10

duration of command

4:213, 4:215

duration of errors 4:217,
4:219

duration of response
4:213, 4:215

distance, compensating for
discrepancies

:probeplan:align

command 4:118

:set:compensation

command 4:153, 4:155,
4:157

DNE bit 3:14, 3:17

DNE bit STB register 3:14

documentation *ii*

additional sources of
information *ii*

command syntax 4:2

online help 1:6

organization *ii*

updates and revisions *iii*

dollar sign, preceding meta
commands 3:30

done bit, STB register see

DNE bit

down

lowering the chuck 4:37,
4:39

download

GPIB probeplan

command 4:122

DUT see device-under-test

Dynamic Data Exchange see

DDE

Dynamic Data Exchange

Management Library see

DDEML

E

echoing

commands 4:200

error messages 4:163,
4:204, 4:206

response messages
4:165, 4:208

- edgesense
 - setting overdrive 4:169
 - setting switch parameter 4:171
 - turning on/off 4:167
- end DDE session 2:11
- end-of-file string 3:29
- English units of measure 4:226
- ERR bit 3:13
- ERR flag STB register 3:14
- error
 - :set:display 4:163
 - :set:show 4:204
 - :system query 4:243
 - returned with
 - :probeplan:\download command 4:123
- error bit, STB register see ERR flag
- error flag 3:13
 - STB register 3:14
- error?
 - :set:show 4:206
- errors
 - @ (at sign) character preceding error messages 3:26, 3:29
 - command-error bit
 - ESR register 3:14
 - duration of display 4:217, 4:219
 - execution-error bit
 - ESR register 3:14
 - flag in STB register 3:14
 - pausing on 3:7
 - recommended meta-command response settings 3:27
 - terminating GPIB Server on 3:7
- ese (event status enable) command 3:33
- ese (event status enable) query 3:33
- ESE register see event-status enable register
 - masking events 3:14, 3:17
- ESR see event status register
- ESR register see standard event-status register
 - masking events 3:14, 3:17
- esr? (event status register) query 3:33
- event status register 3:13
- events
 - masking 3:14, 3:17
 - events generating a service request 3:14
 - event-status enable register 3:14
 - masking ESR register 3:15
- examples
 - string-return handshaking 3:10
- EXE (execution error) bit
 - ESR register 3:14
- execute
 - DDE application file 2:8
- execution
 - of commands 3:27

execution-error bit
 ESR register 3:14
 exit DDE 2:11
 Expect DOS 10K syntax 3:8

F

factor
 setting compensation
 4:153, 4:155, 4:157
 FALSE/TRUE return string
 4:30, 4:39, 4:101, 4:113,
 4:159, 4:161
 fast/slow see velocity
 files
 AUTOEXEC.BAT 1:3
 CONFIG.SYS 1:3
 CSGPIB.INI file 3:35
 deleting a probe plan
 4:120
 downloading a probe
 plan 4:122
 end-of-file string 3:29
 getting path 4:249
 loading probe plan
 4:125
 mistakes in uploading
 3:29
 readme.txt *iii*
 running a DDE
 application 2:8
 uploading a probe plan
 4:147
 win.ini 4:249
 first die moves 4:58
 first subsite moves 4:61
 four characters meta and
 probe-station commands
 abbreviated to 3:30

function call
 to the DDE Server
 2:12–2:14

G

General Purpose Interface
 Bus see GPIB
 General-Purpose Interface
 Bus see GPIB
 generating a service request
 3:14
 GPIB
 :probeplan
 :delete command
 4:120
 :download command
 4:122
 :upload command
 4:147
 \system\display
 command 4:241
 address assignment 3:5,
 3:7
 backward compatibility
 3:8
 command list 3:33
 command responses 3:4
 Command Server 3:6
 command summary
 table 4:3–4:19
 controller 3:3
 error query 4:243
 initialization file 3:35
 instrument 3:3
 instrument-defined bits
 3:14
 meta command set used
 with 3:30

- opc command 3:15
- protocol 3:2
- query responses 3:4
- run in virtual mode 4:232, 4:234
- Server window 3:6, 3:9
- Setup dialog box 3:7
- standards 2:12
- Start! command 3:9
- starting server 3:6
- switch box 3:3
- terminating server session 3:7
- timeout 3:17–3:20
- timeout value 3:8
- timeout values and delays
 - :set:delay command 4:159, 4:161
 - :system:delay command 4:239
- timeout values string-return handshaking 3:27
- troubleshooting 3:35–3:38
- verifying communication 3:8–3:11
- verifying remote communication 3:10
- GPIB commands
 - *stb? 3:10
 - :host 4:173
 - :host? 4:175
- GPIB Server communications model 1:6–1:8

H

- halting stage movements see :move:stop commands
- handshaking 3:27
 - methods of 3:12
 - SRQs 3:26
 - string-return method 3:26, 3:27
 - string-return timeout values for 3:27
- hardware
 - initializing 2:5
 - running DDE without 2:11
 - system requirements 1:2
- help
 - online 1:6
- Help menu 1:4
- high/low see acceleration
- home see also :move:center command
 - moving stage to 4:41
- horizontal stage movements
 - :move:absolute commands 4:19–4:30
 - \move\probeplan commands 4:45–4:79
 - :move:relative commands 4:80–4:88
 - :move:scan:xaxis 4:92
 - :move:stop:xaxis 4:105
- host see also remote communication
 - \probeplan\download 4:122, 4:147
 - address assignment 3:5, 3:7

- address assignment
 - incorrect 3:11
 - program responsibility
 - for handshaking 3:26
 - recommended meta-command response
 - settings 3:27
 - remote communication
 - setup 3:8
 - responses to 3:5
- host connection 4:173, 4:175
- hot link
 - interprocess conversion
 - 2:3

I

- IBIC program 3:8
- icon
 - DDE Server 2:5
- identification
 - device number 4:212
 - query for string 3:33
 - string returned to host
 - 4:245
 - system query 4:245
 - version number 4:252
- identifying communication problems
 - GPIB 3:11
- idn? (identification) query
 - 3:33
- IFC (Interface Clear)
 - command 3:3
- imperfections
 - setting compensation
 - factors 4:153, 4:155, 4:157

- setting probeplan
 - alignment position
 - 4:118
- in/out stage movements
 - :move:absolute
 - commands 4:19–4:30
 - :move:relative
 - commands 4:80–4:88
 - :move:scan:yaxis 4:94
 - :move:stop:yaxis 4:107
- index numbers in probe plans
 - for die sites 4:45, 4:58, 4:59, 4:64, 4:68, 4:74
 - for subsites 4:53, 4:54, 4:56, 4:61, 4:66, 4:72
- initialize
 - probe-station hardware
 - 2:5
- initializing
 - stage 4:41
- instruction sets
 - meta command set 3:27
 - meta commands used, regardless of the protocol chosen 3:30
- instrument 3:3
- instrument-defined bits 3:14
- instruments
 - address incorrect 3:11

- interface
 - interface clear command 3:3
- interprocess conversion
 - cold link 2:3
 - hot link 2:3
 - warm link 2:3
- ISS
 - (Impedance Standard Substrate) see also device-under-test
 - italics, used to identify a replaceable parameter or user-defined value 4:2
- item name
 - string for 2:3
- L**
- last die moves 4:64
- last subsite moves 4:66
- left/right stage movements
 - :move:absolute commands 4:19–4:30
 - :move:relative commands 4:80–4:88
 - :move:scan:xaxis 4:92
 - :move:stop:xaxis 4:105
- light
 - :set command 4:177
 - microscope 4:177
- limit
 - travel along axes 4:221
- literal parameter, boldface for indicating 4:2
- load see also upload and download
 - move command 4:43
 - probeplan command 4:118, 4:125
 - load-device feature
 - :move:load command 3:11
 - load-device feature. See move
 - load command 3:10
 - loading a device. See load-device feature 3:11
 - local operation see manual mode and direct control
 - Local/Load feature 3:11
 - location of stage see :move commands
 - returning 4:23
 - location of the stage
 - load-device position 3:11
 - location of the stage. See also stage-movement commands 3:11
- lock
 - :lock? 4:247
 - :unlock 4:250
 - system command 4:246
- long form of meta and probe-station commands 3:30
- low/high see acceleration
- lowercase
 - commands are not case-sensitive 3:30
- lowering/raising the chuck
 - :move:absolute commands 4:19–4:30
 - :move:down 4:37, 4:39

:move:relative
 commands 4:80–4:88
 :move:up 4:111, 4:113
 :set:acceleration
 commands 4:149–4:150
 :set:velocity commands
 4:230–4:231

M

machine 0,0 stage location
 4:31

manifold

auxiliary power on/off
 4:151
 microscope light on/off
 4:177
 vacuum 4:228

manual

:set mode command
 4:178
 command syntax 4:2

manual?

:set mode query 4:180

manuals *ii*

marked die for testing
 retrieving number of
 4:133
 setting color of 4:130

masking bits 3:15

masking events from
 generating an SRQ 3:14,
 3:17

master-status bit, STB
 register 3:14

measure

units of 4:226

medium see velocity or
 acceleration

menu

Config 3:7

menu interface refer to PCS
 and PPE User Guides

menus

Help menu 1:4

messages see also errors

:set:display commands
 4:165

COMPLETE string,
 OPC or DNE bit
 generating 3:14
 displaying 3:10
 displaying error 4:163,
 4:204, 4:206
 displaying information
 4:165, 4:241

displaying responses
 4:208

end of file 4:122

errors @ (at sign)
 character preceding
 3:26, 3:29

GPIO error 3:4

inserting 4:241

meta commands 3:31–3:32

\$ (dollar sign) preceding
 commands 3:30

\:set responses 3:4

:set:error:abort 3:27

:set:error:pause 3:27

:set:respond 3:27

:set:show:command
 3:27

:set:show:pause 3:27

:set:show:response 3:27

- abbreviated form. See probe-station
- abbreviated syntax 3:30
- complete form. See probe-station complete
- syntax 3:30
- recommended settings 3:27
- SCPI-compatible syntax 3:30
- syntax 3:30
- metric units of measure 4:226
- microscope
 - turning light on/off 4:177
- milliseconds see delay
- motors
 - turning on/off 4:196
- mouse
 - use with DDE and GPIB 1:2
- move
 - load command
 - example 3:10
- move commands *See also*
- move priority*
 - :absolute 4:20, 4:25, 4:59, 4:187, 4:230
 - :absolute:chuck 4:20, 4:25, 4:187
 - :absolute:nchuck 4:28
 - :absolute? 4:23
 - :center 4:31, 4:37, 4:41, 4:111, 4:230
 - :contact 4:30, 4:31, 4:33, 4:37, 4:49, 4:53, 4:58, 4:61, 4:64, 4:66, 4:68, 4:70, 4:72, 4:74, 4:76, 4:78, 4:111
 - :contact? 4:30
 - :down 4:37
 - :down? 4:39
 - :home 4:41
 - :load 4:43
 - :probeplan:absolute:die 4:45
 - :probeplan:absolute:index 4:49
 - :probeplan:absolute:subsite 4:53, 4:54
 - :probeplan:absolute:subsite? 4:56
 - :probeplan:first:die 4:58, 4:59
 - :probeplan:first:subsite 4:61
 - :probeplan:last:die 4:64
 - :probeplan:last:subsite 4:66
 - :probeplan:next:die 4:54, 4:62, 4:68
 - :probeplan:next:site 4:70
 - :probeplan:next:subsite 4:62, 4:72
 - :probeplan:prior:die 4:74
 - :probeplan:prior:site 4:76
 - :probeplan:prior:subsite 4:78
 - :relative 4:80, 4:83, 4:242
 - :relative:chuck 4:80, 4:83

:relative:nchuck 4:86
 :scan:aaxis 4:89
 :scan:xaxis 4:92
 :scan:yaxis 4:94
 :scan:zaxis 4:97
 :separate 4:31, 4:37,
 4:99, 4:111
 :separate? 4:101
 :stop:aaxis 4:103
 :stop:xaxis 4:105
 :stop:yaxis 4:107
 :stop:zaxis 4:109
 :up 4:111
 :up? 4:113
 :wait 4:20, 4:25, 4:28,
 4:114
 :wait? 4:116
 move priority, setting 4:188
 movement commands. See
 stage-movement commands
 3:11
 MSS bit STB register 3:14
 multiple commands on a
 command line, syntax 4:2
 multiple commands on a line
 meta 3:30

N

name
 string for 2:3
 National Instruments GPIB
 interface card and software
 3:8
 nchuck see no-chuck stage
 movements
 next die moves 4:68
 next site moves 4:70
 next subsite moves 4:72

no-chuck stage movements
 :move:absolute:nchuck
 4:28
 :move:relative:nchuck
 4:86
 noise, motor
 reducing 4:196
 notes see messages
 number
 characters meta and
 probe-station commands
 are abbreviated to 3:30
 number of die marked for
 testing 4:135

O

obsolete GPIB commands
 3:8
 ONL bit 3:13
 online
 bit in STB register 3:13
 GPIB Server 3:9
 online help 1:6
 opc (operation complete)
 command 3:33
 OPC bit 3:14, 3:17
 opc? (operation complete)
 query 3:33
 operation-complete
 bit 3:14, 3:17
 operation-complete (OPC)
 bit
 ESR register 3:14
 operation-complete string
 response in string-response
 handshaking 3:26, 3:27
 origin
 changing 4:23

- distance from limits
4:221, 4:224
- moving to 4:31, 4:41
- setting user 4:20, 4:23,
4:187
- overtravel 4:169

P

- palette 4:130, 4:137
- parameters
 - delimited 4:2
 - GPIO remote
communication setup
3:8
 - separating 3:30
 - setting up GPIO 3:6
 - space, used to delimit
commands and
parameters 4:2
 - syntax 4:2
- path query 4:249
- pause see delay or wait
- Pause on error 3:7
- PCS 1:1
 - getting path 4:249
- pointer
 - to messages 2:4
- polling see serial polling
- PON (power-on) bit
 - ESR register 3:14
- position of stage
 - returning 4:23
- positioning the stage see
:move commands
- positioning the stage. See
stage-movement commands
3:11
- power

- turning on/off auxiliary
4:151
- power-on bit
 - ESR register 3:14
- presetting user coordinate
system 4:186
- prior die moves 4:74
- prior site moves 4:76
- prior subsite moves 4:76
- priority of stage moves
 - :set:precise 4:185
 - :set:wafer 4:236
- probe station
 - auxiliary power, turning
on/off 4:151
 - command syntax 4:3
 - compensation factor,
using 4:153, 4:155,
4:157
 - device identification
number (devID
parameter) 4:21
 - distance from center to
stage-travel limit
 - returning 4:221,
4:224
 - how communication
works 1:6–1:8
 - initializing 4:41
 - loading a device 4:43,
4:44, 4:91, 4:228
 - microscope light,
turning on/off 4:177
 - power up state 4:41
 - setting origin for
coordinates 4:20, 4:23,
4:187

- vacuum, turning on/off
4:33, 4:43, 4:44, 4:91
- version 4:245, 4:252
- probeplan commands
 - :align 4:118, 4:125
 - :delete (GPIB only)
4:120
 - :download (GPIB only)
4:122
 - :load 4:118, 4:125
 - :load? 4:128
 - \:mark\:die 4:130
 - :move,:absolute\:die:
4:45
 - :move,:absolute\:index
4:49
 - :move,:absolute\:subsite
4:53
 - :move,:absolute\:subsite?
4:56
 - :move,:first\:die 4:58,
4:59
 - :move,:first\:subsite 4:61
 - :move,:last\:die 4:64
 - :move,:last\:subsite 4:66
 - :move,:next\:die 4:62,
4:68
 - :move,:next\:site 4:70
 - :move,:next\:subsite 4:72
 - :move,:prior\:site 4:76
 - :move,:prior\:die 4:74
 - :nsubsite? 4:133
 - :ntested? 4:135
 - \:set\:palette 4:137
 - :set\:subsite 4:142
 - :show 4:140
 - :unload 4:145
 - :upload (GPIB only)
4:147
- ProbePlan files
 - command responses
3:29
 - end-of-file string 3:29
- Prober Control Software see
PCS
- probe-station
 - identification string
4:245
- probe-station commands
 - abbreviated form 3:30
 - complete form 3:30
 - long form 3:30
 - syntax 3:30
- probe-station control
 - interfaces provided 1:1
- probe-station hardware
 - initializing 2:5
- probing
 - caution for using no-
chuck commands 4:30
 - contacting the DUT
4:49, 4:53, 4:58, 4:61,
4:64, 4:66, 4:68, 4:70,
4:72, 4:74, 4:76, 4:78,
4:91, 4:99
 - die sites 4:62, 4:64,
4:68, 4:76, 4:142
 - separating from the
DUT 4:99, 4:101
 - setting overdrive 4:169
 - subsites 4:53, 4:61,
4:72, 4:142
- problems see communication
problems
- program examples

- C language 2:12
 - Command Server 5:2
 - sample DDE 2:12–2:14
 - Visual Basic 2:15
 - programs
 - responsibility of the host for handshaking 3:26
 - prompts see messages
 - displaying messages 3:10
 - generating sounds 4:238
 - protocols 3:2
 - meta commands used, regardless of the protocol chosen 3:30
 - protocols. see also GPIB, SCPI, and Command
- Q**
- queries
 - recommended meta-command response settings 3:27
 - syntax 4:3
 - query
 - :move:absolute? 4:23
 - :move:contact? 4:35
 - :move:down? 4:39
 - :move:probeplan:absolut e:die? 4:47
 - :move:probeplan:absolut e:index? 4:51
 - :move:probeplan:absolut e:subsite? 4:56
 - :move:separate? 4:101
 - :move:up? 4:113
 - :move:wait? 4:116
 - :probeplan:load? 4:128
 - :probeplan:nsubsites? 4:133
 - :probeplan:ntested? 4:135
 - :salign? 4:199
 - :set:busy? 4:152
 - :set:delay? 4:161
 - :set:host? 4:175
 - :set:manual? 4:180
 - :set:msafety? \i 4:184
 - :set:show:error? 4:206
 - :set:show:response 4:210
 - :set:show:response? 4:208
 - :set:station? 4:212
 - :set:time:command 4:213
 - :set:time:command? 4:215
 - :set:time:error? 4:219
 - :set:travel? 4:224
 - :set:virtual? 4:234
 - :show:command? 4:202
 - :system:error? 4:243
 - :system:identification? 4:245
 - :system:lock? 4:247
 - :system:path? 4:249
 - :system:version? 4:252
 - response 3:4
 - question mark, indicating a query 4:3
 - quiet
 - :set mode command 4:196
 - quit

DDE Server session
 2:11

quitting a remote session
 :set:manual 4:178
 \unlock 4:250

quotes, double
 used in string parameter
 4:241

quotes, single
 used in string parameter
 4:241

R

raising/lowering the chuck
 :move:absolute
 commands 4:19–4:30
 :move:down 4:37, 4:39
 :move:relative
 commands 4:80–4:88
 :move:up 4:111, 4:113
 :set:acceleration
 commands 4:149–4:150
 :set:velocity commands
 4:230–4:231

rate see acceleration or
 velocity

readme.txt file *iii*

registers see status-reporting
 structure
 masking events from
 generating an SRQ
 3:14, 3:17
 status-reporting
 structure 3:14
 verifying register-read
 capability 3:10

remote communication see
 also communication
 problems
 commands requiring
 longer timeout values
 4:159, 4:239
 commands that require
 longer timeout values
 3:20
 delaying command
 execution 4:159, 4:182,
 4:185, 4:239
 GPIB switchbox 4:173,
 4:175
 recommended meta-
 command response
 settings 3:27
 responses to the host
 4:243, 4:245
 safety prompting 4:184
 syntax for commands
 4:2
 troubleshooting GPIB
 3:11–3:12
 troubleshooting GPIB
 communication 3:11
 using serial polls for
 delays 4:239
 using SRQ handshaking
 for delays 4:239
 verifying GPIB
 communication 3:10

replaceable parameter,
 italics for indicating 4:2

request-service bit, STB
 register 3:14

response field
 clearing 3:8

- responses see messages
 - duration of display
 - 4:213, 4:215
 - turning on 3:4
- revision information *iii*
- right/left stage movements
 - :move:absolute
 - commands 4:19–4:30
 - :move:relative
 - commands 4:80–4:88
 - :move:scan:xaxis 4:92
 - :move:stop:xaxis 4:105
- RQS bit
 - STB register 3:14
- RS-232C
 - meta command set used with 3:30
- rst (reset) command 3:33
- run
 - GPIB Server 3:6
- S**
- safety prompting 4:184
- scan commands see also
- stop commands
 - :move,:aaxis 4:89
 - :move,:xaxis 4:92
 - :move,:yaxis 4:94
 - :move,:zaxis 4:97
 - :move: 4:105, 4:107, 4:240
 - for theta adjustment 4:89, 4:92, 4:94
- SCPI
 - meta commands have compatible syntax 3:30
 - standard 3:2
 - standards 2:12
- screen, displaying
 - alignment position 4:118
 - commands 4:200, 4:202, 4:213, 4:215
 - errors 4:163, 4:204, 4:206, 4:217, 4:219
 - load dialog box 4:44
 - messages 4:165, 4:241
 - responses 4:208, 4:210, 4:215
- seconds see delay
- semi-automatic chuck moves
 - setting acceleration 4:149
 - setting velocity 4:230
- semi-automatic moves
- timeout values affecting handshaking 3:20
- semicolon separating multiple commands 3:30
- semicolon, separating commands 4:2
- serial polling 3:9, 3:20
 - DNE bit 3:14, 3:17
 - method of handshaking 3:12
 - OPC bit 3:14, 3:17
 - STB register 3:13
- serial polls see display and delay
- server
 - defined 2:2
- server application name
 - string for 2:3
- service request see SRQ
- service-request enable register 3:14

service-request generation

see SRQ

set commands

:acceleration 4:103,
4:149
:auxiliary 4:151
:busy? 4:152
:compensation 4:153
:compensation:factor
4:155
:compensation:mode
4:157
:delay 4:159, 4:239
:delay? 4:161
:display:errors 4:163
:display:messages 4:165
:edgesense 4:167
:edgesense:overdrive
4:169
:edgesense:switch 4:171
:host 4:173
:host? 4:175
:light 4:177
:manual 4:178
:manual? 4:180
:msafety 4:182
:msafety? 4:184
:precise 4:185
:preset 4:20, 4:23, 4:25,
4:186
:priority:accuracy:modul
e 4:188
:priority:accuracy:wafer
4:190
:priority:standard:modul
e 4:192
:priority:standard:wafer
4:194

:quiet 4:196
:salign 4:197
:salign? 4:199
:show:command 4:200
:show:command? 4:202
:show:error 4:204
:show:error? 4:206
:show:response? 4:208,
4:210
:show:time:command
4:213
:show:time:command?
4:215
:show:time:error 4:217
:show:time:error? 4:219
:station? 4:212
:subsite 4:53, 4:61
:travel. 4:221
:travel? 4:224
:unit 4:186, 4:226
:vacuum 4:228
:velocity 4:230
:virtual 4:232
:virtual? 4:234
:wafer 4:236
probeplan palette 4:137

set commands:error:abort
command 3:27

set commands:error:pause
command 3:27

set commands:respond
command 3:27

set
commands:show:command
command 3:27
set commands:show:pause
command 3:27

- set
 - commands:show:response
 - command 3:27
 - setting subsite locations 4:142
 - setting user coordinate
 - system 4:186
 - in absolute moves 4:19–4:30
 - in relative moves 4:80–4:88
 - specifying units of measure 4:226
- settings
 - bits DNE bit 3:14, 3:17
 - GPIB 3:6
 - meta commands
 - recommended settings 3:27
- setup
 - DDE parameters 2:8
 - GPIB parameters 3:5
- setup commands
 - :set:acceleration 3:20
 - :set:delay 3:20
 - :set:velocity 3:20
- show commands 4:200, 4:202, 4:204, 4:206, 4:208, 4:210, 4:213, 4:215, 4:217, 4:219
 - probeplan 4:140
- slow/fast see velocity
- software
 - alignment mode 4:197, 4:199
 - Command Server 5:2
 - DDE virtual mode 2:11
 - GPIB IBIC program 3:8
 - starting DDE Server 2:5
 - system requirements 1:2
 - Windows API 2:12
- software interface
 - MS Windows 1:2
 - types provided 1:1
- sound, beep command 4:238
- space used to separate commands and parameters 3:30
- space, used to delimit commands and parameters in a command line 4:2
- speed see velocity
- sre (service request enable) command 3:33
- SRE register see service-request enable register
- sre? (service request enable) query 3:33
- SRQ
 - generating a service request 3:14
 - masking events from generating an SRQ 3:14, 3:17
 - preventing 3:16
 - status-reporting structure 3:14
 - using 3:23

- stage
 - moving. See stage-movement commands
 - 3:11
- stage movements see also
- :move commands
 - setting alignment
 - location for probe plan 4:118
 - setting origin for 4:23, 4:31
 - using compensation factors 4:153, 4:155, 4:157
 - verifying 3:10
- stage-movement commands
- :move:load 3:11
- Standard Commands for Programmable Instruments
- see SCPI
- standard event-status register 3:14
 - masking 3:15
- standards 3:2. see also GPIB, SCPI, and Command SCPI, meta commands have compatible syntax 3:30
- Start! command 3:9
- station see probe station. see probe station
- status byte
 - serial polling 3:9
- status byte register 3:13
- status byte. See STB register or *stb? query 3:10
- status of operation
 - :set:manual? 4:180
- status-byte register 3:14
 - masking 3:16
 - reading 3:16
- status-reporting structure 3:2, 3:13–3:17
 - bits defined by instrument 3:13
 - clearing response 3:8
 - commands controlling 3:33
 - conventions in describing 3:13
 - masking bits 3:15
 - masking events 3:14, 3:17
 - verifying register-read capability 3:10
- STB see status byte register
- STB register
 - reading, example 3:10
- stb? (status byte) query 3:33
- stop commands
 - :move 4:89, 4:92, 4:94, 4:97, 4:103
 - :move, aaxis 4:103
 - :move, xaxis 4:105
 - :move, yaxis 4:107
 - :move, zaxis 4:109
- string
 - application name 2:3
 - Command Server 5:2
 - COMPLETE 2:4, 3:4
 - COMPLETE string, OPC or DNE bit generating 3:14
 - DDE command 2:13
 - DDE query response 2:4
 - identification 3:33

- item name 2:3
- pointer to
 - error message 2:4
 - quotes in 4:241
 - topic name 2:3
 - used in :system:display command 4:241
- string-return
 - method of handshaking 3:12
- string-return handshaking 3:26, 3:27
 - IBIC program 3:8
 - timeout values 3:27
- strings
 - end-of-file string 3:29
 - responses string-return handshaking 3:26, 3:27
- subsites
 - caution for defining 4:53, 4:61
 - moving to 4:51, 4:53, 4:56, 4:61, 4:66, 4:72, 4:78
 - setting 4:142
- substrate see ISS
- Summit
 - instrument-defined bits 3:14
- switch
 - :GPIB switchbox 4:173, 4:175
 - edgesense probe 4:171
- symbols
 - preceding GPIB commands 3:3
- syntax
 - abbreviated commands 3:30
 - abbreviating commands 4:3
 - Command Server
 - conformance to standards 2:12
 - complete form of 3:30
 - DDE conformance to standards 2:11
 - long form of meta and probe-station commands 3:30
 - meta commands 3:30
 - probe station commands 4:2
 - probe-station commands 3:30
 - queries 4:3
- system commands
 - :beep 4:238
 - :delay 4:239
 - :display 4:241
 - :error? 4:243
 - :identification 4:245
 - :lock 4:246
 - :lock? 4:247
 - :path? 4:249
 - :system:display 3:10
 - :unlock 4:250
 - :version? 4:252
 - display command 3:10
- system requirements 1:2

T

- technical support 1:3
- temperature

- using compensation factor 4:153, 4:155, 4:157
- Terminate on error 3:7
- termination characters
 - end-of-file string 3:29
- test environment 3:2
- testing
 - returning die marked for 4:135
- thermal controller
 - controlling remotely 3:3
- theta adjustment see :move:scan commands
 - software alignment mode 4:197, 4:199
 - with scan command 4:89, 4:92, 4:94, 4:97
- time see delay or timeout values
- timeout value 3:8
- timeout 3:8
 - alternatives to 3:20
 - GPIO 3:6, 3:17–3:20
 - setting GPIO 3:7
- timeout value
 - string-return handshaking 3:27
- timeout values
 - commands requiring longer 4:159, 4:239
 - commands that require longer values 3:20
 - setting 3:20
- topic
 - string for 2:3
- topic name
 - string for 2:3
- travel
 - query: 4:224
 - setting 4:221
- troubleshooting
 - GPIO 3:11–3:12, 3:11–3:12, 3:35–3:38
 - technical support 1:3
- TRUE/FALSE return string 4:30, 4:39, 4:101, 4:113, 4:159, 4:161
- tst? (self-test) query 3:33
- U**
- ultra fast/ultra slow see velocity
- unit
 - :set command 4:186, 4:226
- units of measure see unit
- unload
 - probeplan 4:145
- unlock
 - :system command 4:250
- up/down see chuck
- update information *iii*
- upload
 - GPIO probeplan command 4:147
- uploading files
 - end-of-file string 3:29
 - mistakes 3:29
- uppercase
 - commands are not case-sensitive 3:30
- user coordinate system see coordinate system

- user interaction affecting timeout value requirements 3:20
- user-defined messages
 - displaying 3:10
- user-defined messages, displaying see display and messages
- V**
- vacuum
 - :set command 4:228
 - with move commands 4:31
- values
 - timeout 3:6
- variables
 - device 3:13
- velocity
 - :set:command 4:230
 - timeout values affecting handshaking 3:20
- verifying GPIB communication 3:8–3:11
- verifying GPIB remote communication 3:10
- verifying remote communication
 - GPIB 3:8–3:11
- version
 - firmware release 4:252
 - query 4:252
- vertical chuck movement see also chuck, raising and lowering
 - :move:down 4:37
 - :move:down? 4:39
 - :move:up 4:111
 - :move:up? 4:113
 - setting acceleration 4:149
 - setting velocity 4:230
 - stopping 4:109
 - zaxis 4:97, 4:109
- very fast/very slow see velocity
- very high/very low see acceleration
- view through microscope
 - probe movement note 4:90, 4:93, 4:95
- virtual
 - :set mode command 4:232
 - :virtual? 4:234
- virtual mode
 - operation of DDE 2:11
- virtual?
 - :set mode command 4:234
- Visual Basic
 - DDE program example 2:15
- VNA Calibration software 3:3
- W**
- wafer
 - :set command 4:236
 - making contact with probes 4:33, 4:99, 4:101
 - subsites
 - caution for defining 4:61
 - defining 4:142

- theta-adjustment 4:89,
4:92, 4:94
- wafer map
 - colors in 4:130, 4:137
 - controlling display of
4:140
- wafer mode
 - setting height 4:221
- wait see also delay
 - :move:wait command
4:114, 4:145
 - :move:wait? command
4:116
 - with other commands
4:20, 4:25, 4:28, 4:114,
4:116
- warm link
 - interprocess conversion
2:3
- warning see caution
- window
 - DDE 2:5
 - GPIB 3:6
 - GPIB Server online 3:9

Z

- z-axis see also stage
movements and vertical
chuck movements
 - absolute moves along
4:19–4:30
 - relative moves along
4:80–4:88
 - scanning along 4:97



MP MASTER



MA32764